Imperial College London

BEng Individual Project

Imperial College London

Department of Computing

Improving Scrabble Play Using Quackle

Author:
Mihey Matev

Supervisor:
Prof. Thomas Lancaster

Second Marker:
Dr. Mark Wheelhouse

September 21, 2022

Abstract

Scrabble is a game in which players take turns in placing down words that they form from their rack of seven letter tiles on a 15 by 15 board. Each word they form increases their score. After each play, they refill their rack by drawing new random letter tiles from a shared bag. The game ends once the bag and any one player's rack is empty. The player with the highest score at the end wins.

Scrabble is a game of strategy, word memorisation, and randomness. The goal of this project is to present strategic move choices to players in an intuitive, memorable way. As such, the negative effects of randomness caused by tile drawing can be decreased, and lead to higher win rates. This report describes the development of three AIs, each of which provides hints to the user about the best moves to make which maximise their chances of winning.

Artificial intelligence (AI) players are used to generate strategic choices or 'hints' and present them to the user: Championship player, Static player and Greedy player. Greedy plays the currently highest-scoring word. Static only considers the current game position, and based on the move considered and the letters left after it ('rack leave'), gives a heuristic score ('equity' or 'valuation') to the move; then, it plays the move with the highest equity. Finally, Championship builds upon Static: it takes the top moves sorted by equity and simulates playing them to see which gives it the best future position.

The Greedy AI presents the top-scoring moves as hints. A meaningful message is associated with each calculation of the Static AI. Since Championship player runs hundreds of simulations to reach a result, it is not feasible to explain each simulation to the user, as the user is not expected to manually run hundreds of simulations. However, its output moves can still help the user gauge how good another move is by showing that move's ranking according to Championship AI.

Since Static wins 40% of matches against Championship, which puts them within 100 Elo rating of each other, the hint generation is good enough even if done by Static and not by Championship. Another advantage of using Static over Championship is that the hints are generated instantly, whereas Championship can require waiting upwards of 30 seconds. This makes Static have a good balance of winning moves, hint generation and processing time.

When users were asked to evaluate how useful the hints are, they said that by using this program, they feel that they would be able to improve faster than if they didn't have access to the hints. Also, since Static's hints explain where the suggested moves come from, it's easier to make a mental note of those reasons and evaluate moves based on those.

Contents

Chapter 1

Introduction

1.1 Scrabble Rules

Scrabble is a board game for two to four players created in 1938 by Alfred Mosher Butts. It consists of a 15 by 15 board as pictured in 1.1, a bag of 100 letter tiles (referred to as just tiles from now on) of varying numbers, and player racks. Each tile has an associated score value with it - for example, 'J' has a score of 8, and 'E' has a score of 1. Some tiles are more common than others - for example, there are a total of 3 'L' tiles, however, there are a total of 9 'E' tiles. At the start of the game, each player's rack is populated with the maximum number of 7 tiles drawn randomly from the bag.

After this, each player takes a turn to play. A play can be one of three things: a pass, an exchange, or a placement. A pass means that you skip your turn. An exchange allows you to swap tiles from your rack with random tiles from the bag. You may swap at most 7 tiles or the number of tiles which remain in the bag, whichever is lower. This means that if the bag is empty, an exchange can no longer be made.

Finally, the most interesting and important turn type is placement. The player whose turn it is can use tiles from their rack and place them onto the board spaces to form a word. The player is only allowed to place their tiles either horizontally in the same row (left-to-right) or vertically in the same column (top-to-bottom) in a single-turn; any tile which is placed down must be adjacent to another tile on the board (or be on the central space for the first player to place down a tile), and any string of connected letters must make a valid word. The player who places a word down on the board adds to their score points equal to the sum of the scores of each new word, as seen in 1.2. On top of this, a player can gain an extra 50 points if they use exactly 7 of the tiles from their rack. This is known as a 'bingo'. At the end of a placement turn, if there are tiles left in the bag, the player must replenish their rack by randomly drawing new tiles from the bag until they have 7 or there are no more tiles in the bag.

The game ends once any player runs out of tiles on their rack. At this point, every other player subtracts from their total score the sum of the scores of the tiles on their rack, and the player who used up their whole rack gains additional points equal to the sum of all other players' tile scores. The winner is the person with the highest score (not necessarily the person who was first to empty their rack).

1.2 Foci of Project

For this project, we will focus on two-player Scrabble games in English. Specifically, American English NWL2018 ('NASPA Word List 2018 Edition', where NASPA is 'North American Scrabble Players Association') lexicon is used. Expanding to other languages or lexica should hopefully require little effort, however, expanding to more players might not be as easy. This is because when there are no more tiles in the bag, in a two-player game, both players know the other's rack, whereas in a three- or four-player game, no player can be certain of any other player's rack. This can lead to difficulty in strategising.

The main aim of this project is to implement an AI (artificial intelligence) which can play Scrabble 'optimally', and use its move generation algorithm to give hints to a player on how they can improve

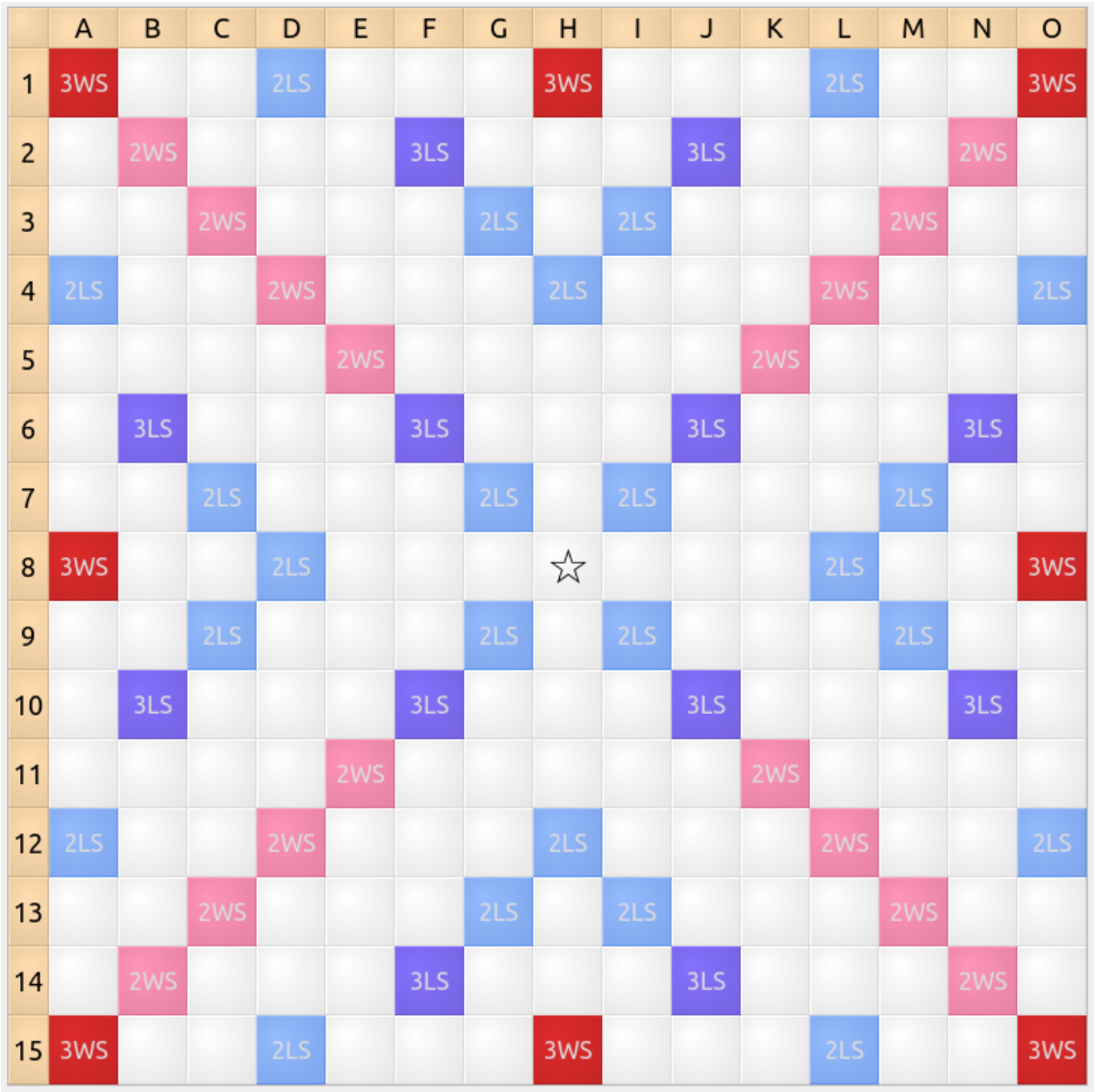

Figure 1.1: A typical Scrabble board layout; the coloured (non-white) spaces represent score multipliers - 2LS doubles the score of the letter on it the first time it is used, 3WS triples the score of the word(s) played using it the first time it is used. The very first play must cross the central tile, hence the special indication with a star.

their own moves. These hints have to be meaningful to a user - they can't just be numbers or seemingly random. The player should be able to find and learn patterns based on the hints so that even when the hints are taken away, they can recall those patterns or calculations to play a good move.

Currently, there seems to be lacking research in the area of helping people to learn how to become better at playing Scrabble. There seems to be little to no research into how to gamify the process of becoming a better Scrabble player. The approach taken here is that the best way to learn something is through doing it and doing it in a guided learning style. This would still involve playing the game multiple times, however, it is expected that by interacting and using the generated hints, the user can learn faster than without them.

Although to win a casual Scrabble game, a good enough strategy for most matches would be to just play the word which gives one the highest amount of points, at a professional level, this is not enough. At a professional level, one has to strategise and think ahead. This is because a player at the championship

Figure 1.2: The first player played 'YOOF' and scored $4+1+1+4=10$ for it; the second player is about to play 'OW' (horizontal), 'YO' (vertical) and 'OW' (vertical) which gives them $(1+2\times4)+(4+1)+(1+2\times4)=23$ points. Note that the multiplication by 2 of the score for the 'W' tile because it is placed on top of a 2LS space which doubles the score of the tile placed on it the first time it is used.

level probably has about a 75% chance of winning against this greedy strategy[1]. One has to save tiles for a bingo, block the opponent from making high-scoring moves if unable to do so themselves, think about the probability of having good tiles left for the next play. This type of long-term thinking and strategising is difficult to learn quickly. This project hopes to make this process of learning faster. As a byproduct, the hope is that the lexicon of the users of the software will also grow.

The issue will be tackled by looking at various Scrabble AIs which can play at championship levels (e.g. MAVEN, Quackle) and analysing their plays to tell the user why that play is 'optimal'. These plays can then be compared to some less optimal plays such as 'play the highest-scoring word possible', which would be a greedy approach, but not always optimal.

[1] https://github.com/michael1212-creator/quackle/blob/simulations/quacker/Champ_v_Greedy_results/results

Chapter 2

Background

2.1 Top Scrabble AI Comparison - Quackle vs. MAVEN

The two most well-known Scrabble AIs are MAVEN by Sheppard (2002, [14]) and Quackle by Jason Katz-Brown et al. ([3]).

Another, more primitive Scrabble AI is described by Appel and Jacobson (1988, [1]) (from here referred to as the 'Greedy' AI). However, it is much poorer than Quackle and MAVEN. This is because the paper mainly focuses on how to implement and use a DAWG (directed acyclic word graph) data structure efficiently, as opposed to playing Scrabble well. All it does is generate all of the possible plays for a turn, and play the highest-scoring one. The problem with this strategy is that there is no strategising - there is no thinking about blocking the opponent from making a valuable play, no thought about improving the current rack for a bingo etc. This does not make a very interesting AI to make use of in the project, as the only hint would be to tell the player to improve their lexicon. Although such an implementation of an AI could help the user get better at playing Scrabble simply by improving their lexicon, it does not generate much interest.

MAVEN is older and has not received any public updates since 2001. There also does not seem to be any publicly available source code for its implementation that one could use to investigate the behaviour of the AI in different situations.

Quackle on the other hand had its last update in 2019, with one of the additions being "Made the simulation engine multi-threaded" ([3]). This is "definitely a lot faster" than just using a single core, which we assume would be the case with MAVEN since there is no mention of anything different. On top of this, it allows the usage of a GADDAG ('directed acyclic graph' prefixed with its own reverse) data structure developed by Steven A. Gordon (1994, [5]). This is the alternative to DAWG, which is what MAVEN uses for its word generation. Although this makes it "nearly five times larger than the DAWG", it "generates moves more than twice as fast" ([5]).

The way MAVEN works is by firstly generating all possible word placements (using DAWG - see discussion below) on the board. For efficiency, at the end of a turn, not all word placements need to be recalculated - only those affected by the newly placed down word. This helps speed up the process. After this, there is a step to evaluate how valuable the letters left on the player's rack after playing any of the possibly playable words are. The rack leave not being valuable enough can cause problems later down the line, especially in an end-game scenario, as it limits the words that can be placed down. So, there has to be a forced way to increase the value of the rack leave, as otherwise due to the nature of good tiles being used up more frequently, a bad rack leave will be favoured. The scoring has to be maximised over longer gameplay as opposed to a single-turn play. The value of a rack leave is found by using a look-up table, which is generated by "a day's worth of self-play games" ([14]), with an initial guess of 0 for every rack.

The reason behind the size increase in GADDAG over DAWG is because, in GADDAG, one word is stored as many times as the number of letters the word contains. This averages out to about five times more memory usage due to the average word length being five letters long. The reason it is faster is that

the algorithm we can use to form words using GADDAG is "more deterministic" ([5]) than that used with DAWG. What is surprising is that although the MAVEN paper by Sheppard (2002, [14]) was last updated in 2002, after the GADDAG data structure implementation in 1994 by Steven A. Gordon ([5]), MAVEN was not optimised to take advantage of it.

Quackle isn't as specific about the player's rack. The player's rack leave is evaluated by adding up the valuation of each letter, the valuation of pairs of duplicate letters, and the valuation of pairs of non-duplicate letters. For example, the tile 'Q' is not a favourable tile to hold, so it is assigned equity of -7.5409 ([1]). However, when combined with a 'U', the pair has a valuation of 13.66 ([2]) which heavily offsets the disadvantage of holding a 'Q'. This overall means that the lookup table for rack worths in Quackle does not have to be as large as in MAVEN, however, it could take more time to do all of the adding up, due to having to reference multiple lookup tables for one value. In the end, the Quackle code implements a lookup table specific to each rack similarly to MAVEN, however, the code allows us to overwrite this.

In MAVEN, there is also the consideration of the tiles left in the bag and their value when combined with the rack leave. Sometimes the tiles left in the bag are worse than a certain rack leave, so MAVEN is less likely to want to play its tiles. Other heuristics used as "vowel/consonant balance" - having a bonus for a good balance of vowels and consonants; "U-With-Q-Unseen" - holding a 'U' with an unseen 'Q' potentially limits the opponent if they are the ones to hold the 'Q', as a word containing the letter 'Q' is highly likely to need a 'U'; "First-Turn Openness" - if MAVEN is the one starting, it could be better to play a short starting word as to limit the opponent from a double-word square.

Sadly, Quackle only considers what is left in the bag near the end of the game. It has no inference engine to allow it to remember which tiles are in the bag or on the opponent's rack due to being exchanged or due to what the opponent played. This potentially makes MAVEN better than Quackle. According to Richards and Amir (2007, [13]), "opponent modelling adds considerable value to simulation". This means that if the AI can consider what the rack of the opponent is (and hence which tiles are left in the bag), there can be an improvement in how well the AI plays. During the simulation phase, Quackle assumes the opponent's rack to be randomly drawn from the unseen tiles. This fails to consider their last played word, which gives information about their rack and hence can allow for higher accuracy simulations. It could even allow the player to think about how to block the opponent from playing high-scoring words or allow them to be more aggressive. The modelling in Richards and Amir's research is also done using the Quackle code, however, at the point when the simulator assumes the opponent's tiles, their gives the AI 1-7 tiles as already being known. Based on this data and Bayes' formula, the researchers then generate the likelihoods of the opponent's leave given a certain play. This allows them to alter the way the opponent draws tiles from the bag to be more like reality, as it increases the certainty that the opponent holds a certain rack. Their conclusion that the "difference is statistically significant with $p < 0.045$" since the score is 5.2 points higher for the more knowledgeable AI does not contradict the simulation results[3], where we do not observe such a drastic score difference, even though there is an even bigger difference in win percentage. However, it would have been more ideal if they talked in terms of average score per turn, as it is a better measure of a player's skill, especially if there is a large imbalance between how well the two players can play (Sheppard 2002, [14] in section "10.1. Skill metrics").

MAVEN splits a game up into three stages: normal game, pre-endgame ("when there are 16 unseen tiles" [14]), and endgame. The former two employ a one-ply simulation, basically simulating a possible situation after any of the words in consideration are played with many caveats in the pre-endgame. In the endgame, since it is a game of perfect information, the "B* search algorithm" is used to find the optimal play. Interestingly, the only factor used in the valuation of the board's state is the availability and ease of access of triple word squares. Note that MAVEN does not take into consideration multiple factors to give a different value - for example, the current state of the board being (or not being) favourable to the tile 'W' would not affect the fact that 'W' is usually a bad tile to hold, and hence the valuation of holding a 'W' is still a negative one.

Quackle is similar to MAVEN in these regards: it also splits the game into three stages, however, the pre-endgame stage is triggered when there are 9 unseen tiles (2 in the bag) as opposed to 16. This

[1] https://github.com/quackle/quackle/tree/master/data/strategy/default_english/worths - on line 17

[2] https://github.com/quackle/quackle/tree/master/data/strategy/default_english/syn2 - on line 314

[3] results from unaltered version of Championship vs. Championship simulations: https://github.com/michael1212-creator/quackle/blob/simulations/quacker/Champ1_v_Champ2_results/results

could make it faster, but could also potentially make it run fewer simulations closer to the end of the game when there is more knowledge, and so miss out on the higher score. However, because before the end-game the unseen tiles are not considered in calculations, increasing this number would probably not have a large effect on the AI's performance. Once again, if a statistical inference engine were used, somewhat like in MAVEN, this could improve Quackle's performance.

Also just like with MAVEN, Quackle does not consider higher-order factors.

MAVEN uses a 1-ply simulation, and Quackle uses a 2-ply simulation. However, diving deeper, we can see that these two mean the same thing. Confusingly, what MAVEN considers as 1-ply, Quackle considers as 2-ply - Quackle counts a ply as a complete cycle, whereas Quackle counts it as 'how many players play after this move', which in a two-player scenario, makes a MAVEN 1-ply equivalent to a Qackle 2-ply. The reasoning behind this is in MAVEN is that "an average turn involves 4.5 tiles", i.e. it would take more than two turns for a rack to completely change on average. The paper argues that deeper ply simulations "are not worth doing", especially since "each iteration takes twice as long" the deeper number of plies used. This was written in 2001, and even with 2019's CPU performance improvements, Quackle seems to consider this many plies as good enough.

In the end, MAVEN plays the word which gives it the highest score across all heuristics on top of the points scored from playing the word itself. Note that the score from the heuristics can be negative. This is very similar to how the Static AI from Quackle generates its moves, with the only difference being that Static does not have a simulation stage.

With DAWG by Appel and Jacobson (1988, [1]), we have to potentially try forming a new word starting at every row and column entry around a pre-existing letter on the board, and then hope that it eventually at least connects up to the already placed letter on the board. This could be up to 81 starting positions we need to test (see 2.1 for possible word starting positions). Then, for every starting position, we need to try all words which contain the black tile's letter. In contrast, GADDAG works by 'hooking' onto the already placed tiles. The starting point is always the placed down tile. We never have to guess whether a word will be able to make use of the already placed tile, as the GADDAG data structure allows us to start our word formation from any letter of a word (as opposed to always having to start it from the first letter of the word in DAWG).

Looking at 2.1, we can see that the way that Quackle works is in some sense similar to MAVEN. Firstly, all possible moves are formed for the current board and rack and similarly updated each turn as in MAVEN, only where needed. This is done using a GADDAG approach outlined in Gordon's 1994 paper ([5]) as opposed to the DAWG approach from [1].

In the simulation stage, Quackle takes a 2-ply approach. This means that the various outcomes are simulated for the next two plays into the future. These are scored based on the score each word gives along with the score for the rack leave (which is once again a precomputed table lookup), after which the top equity scorers are run through simulations to determine the best among the best and eliminate the rest. 2.2 shows the steps Quackle takes to generate its moves.

In contrast to MAVEN, Quackle does not play the word which would make it score the highest amount of points from all of the valuations. Instead, Quackle estimates the percentage of winning from each simulation and chooses to play the word which has the highest percentage to lead to a win. The win percentage estimation is done based on the player's score, tiles left, and 2-ply simulation outcome.

Something which is brought up by Di Maria and Strade (2012, [11]) but seems to be lacking in both Quackle and MAVEN is a discussion on when the AI has a score lead, trying to keep the lead as opposed to trying to be aggressive. There does not seem to be an incentivisation for the AI to be defensive. In Sheppard's 2002 paper ([14]) under the "Making MAVEN stronger" section, the "tenth avenue" mentions how MAVEN prefers to be aggressive over defensive, which lead to the opponent "playing a monster bingo off of it". "Monster bingo" here refers to a move that uses all seven tiles from a player's rack to get a bingo on top of hitting multiple multiplier spaces on the board.

Quackle seems to be the better choice for the AI which should be used to generate the 'optimal' moves which can help the player improve. The open-source nature under the GNU-3 license is another big incentive (as opposed to the closed-source nature of MAVEN, whose rights were purchased by Hasbro according to Richards and Amir, 2007, [13]), as it means the focus of this project can be more on how to generate more useful hints for the player, as opposed to creating an AI. There is some data to suggest

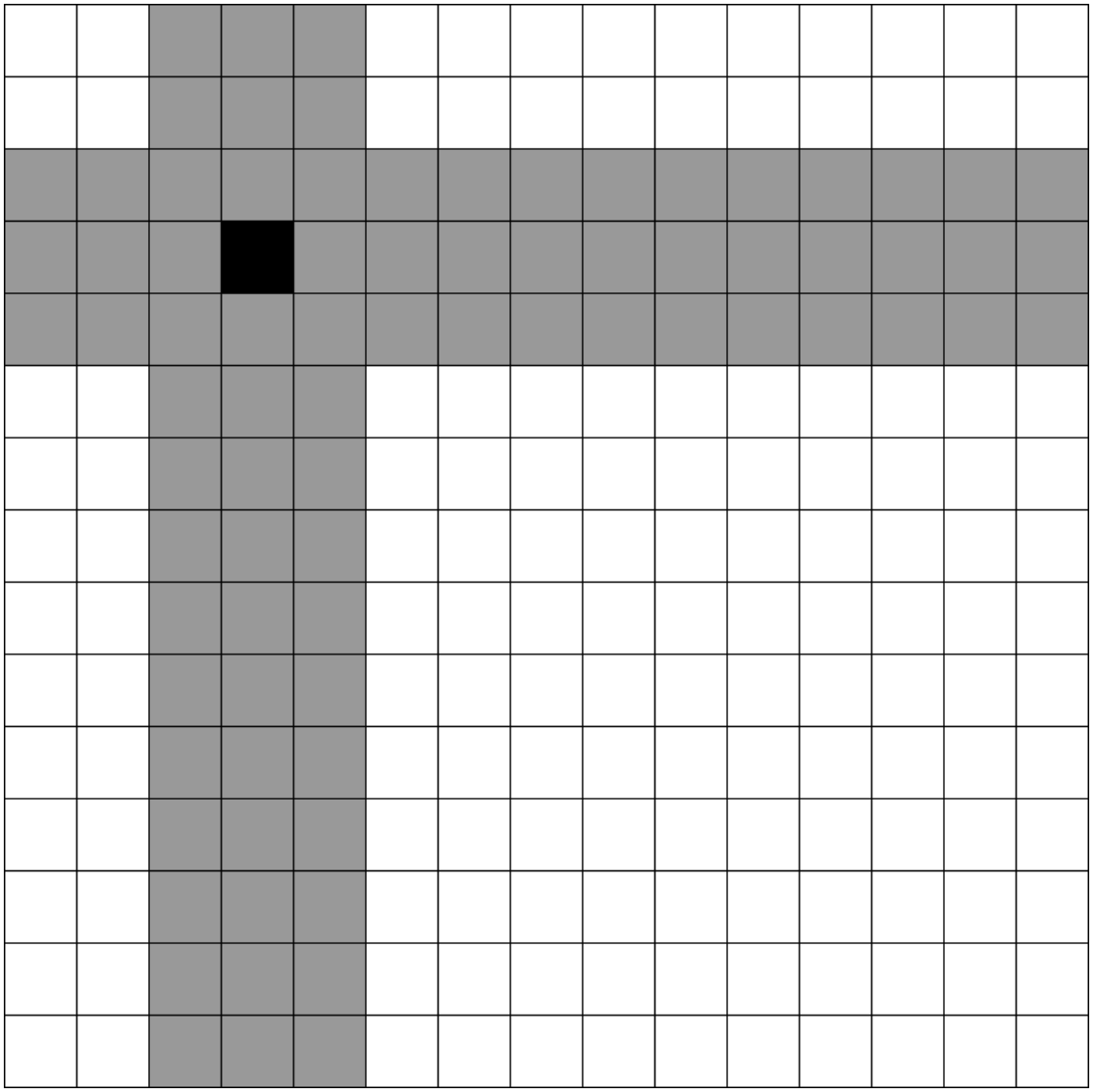

Figure 2.1: Here we can see all of the possible locations we might need to consider starting a word from. Black represents the tile which is already placed down, and grey are the other possible starting locations.

that according to Elo ranking, Quackle is likely to beat MAVEN 85% of the time ([10], [9]). However, as discussed previously, it would be nice to be able to show the player multiple 'optimal' plays from various AIs.

2.2 Other Scrabble AI Heuristics

Some notable alterations to the heuristics of MAVEN can be found in Di Maria and Strade's 'An artificial intelligence that plays for competitive SCRABBLE' (2012, [11]). It suggests using the concrete formula "$F = s - p + c$" with 's' being the points scored by playing the word, 'p' a penalty score, and 'c' a probabilistic "combo score" to account for the possibility of a bingo. The penalty score is suggested to prevent the AI from wanting to open up possible triple word, triple letter, double word, and double letter squares to the opponent, i.e. to make the AI more "defensive". The penalty score sounds useful in theory, however the results of Sheppard's 2002 paper ([14]) state that except for trying to block the opponent from a triple word square, the other bonus squares do not seem to affect the scoring much. However, the probabilistic bingo scoring in the formula might be quite a useful feature in a formula such

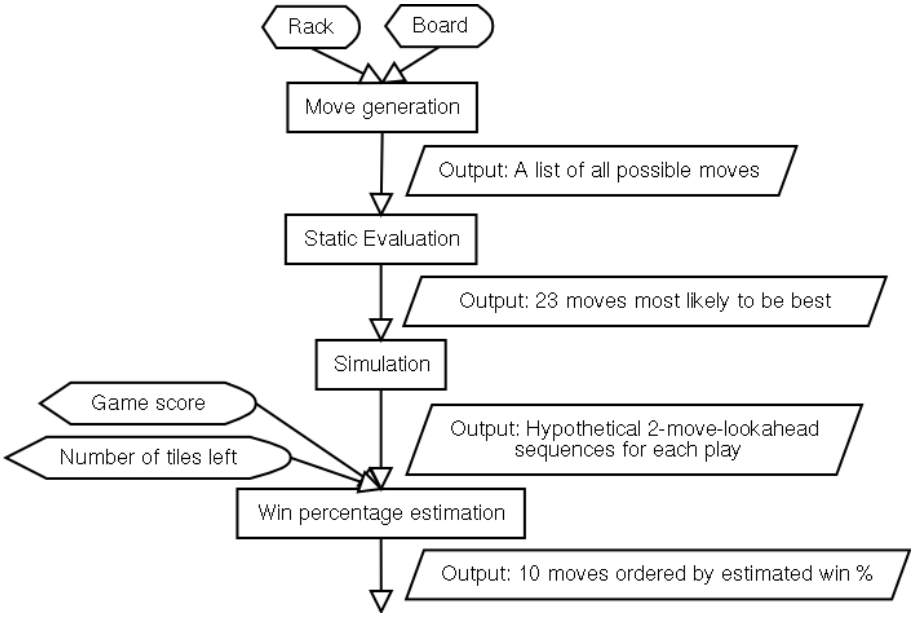

Figure 2.2: diagram of how Quackle chooses its plays. ([6])

as that, as bingos can greatly increase a player's score.

Ramírez et al. in 2009 ([12]) along with describing how to use a different, faster method to generate all possible moves (but using more memory to do so), also has a suggestion which neither Quackle nor MAVEN use: changing the heuristics to be probabilistic. It also uses a similar formula to Di Maria and Strade's of "$v = j + p \times b - d$" for determining the value of a play, where j is the number of points scored by the play, p is the probability of obtaining a bingo after drawing a set number of letters from the bag, b is the expected value of a bingo (e.g. the score a bingo achieves averaged over many games), and d is a defensiveness rating of a move. However, it instead proposes using Monte-Carlo simulation when there is no possibility of a bingo, or otherwise using a probabilistic heuristic for the likelihood of bingo tiles. It uses opponents' modelling in either case, meaning that, unlike MAVEN or Quackle, it might attempt to play aggressively when behind, or defensively when ahead.

Romero et al. (2009, [4]) discuss different heuristics which a human, without having the luxury of being able to make hundreds of simulations, can use. Just like this research, they assume that the player is competent enough to be able to form most, if not all playable words on the board given a rack. They use the formula of "$v = j + p \times b - d$" as their basis, similarly to the one given by Di Maria and Strade (2012, [11]). This formula is relatively easy to evaluate for a human, especially as through simulation some average values for the expected value of a bingo (being 77) can be found. It does however require the player to give an estimate of the defensiveness of a move "d" and the probability of a bingo "p". The paper suggests "value of d is 20... if one puts a tile on the edge allowing a possible nine-timer", "10 if one plays a vowel next to a premium square allowing a six-timer", and "5 if one plays allowing a possible four-timer", with varying values in between. The most difficult part is calculating 'p'. This requires calculations using combinatorics and considering which letters could be drawn to have a rack which results in a bingo. The proposed formula for this is

$$\sum_{\text{all bingos}} \frac{\prod_{\text{tile type}} C(\#\text{ of tiles of this type left in bag}, \#\text{ of tiles of this type required to bingo})}{C(\#\text{ of unseen tiles}, \#\text{ of tiles used in play})}$$

where $C(a, b)$ is the number of ways we can choose b elements out of a set of a elements. This formula fails to account for the probability of drawing a blank tile, which does not seem to be addressed in the paper. Accounting for it would probably make this formula even less feasible for a human, and it is unlikely to make a huge difference to the outcome. Perhaps if values for p could be estimated by simulating many games, an average of them could be presented to the user. This is unlikely to be a great strategy, however, as a bingo can make a very big difference to a game, and estimating both p and

b doesn't account for the large variation of how the game might turn out. A better way to present the value of p to the user might be by letting them play many games with a hints generator that precomputes this value for them. This can allow them to have a better intuition for what this value is based on their rack and unseen tiles the longer they play.

The above heuristic is referred to as the 'default' heuristic and is the one which wins the highest number of times according to Romero et al. (2009, [4]). The 'MaxDouble' heuristic is very similar to the greedy approach of playing the best move on the current turn, however, it does it for two moves. This is potentially quite easy for a human player to implement. Since two moves are considered, and the rack turnover rate is on average 4.5 tiles per turn, it means that the words considered might just be good enough. We can see that it is the worst of the three heuristics. The 'MaxCare' heuristic is not great, as it does not address gameplay before the pre-endgame. Even then, it might be too defensive, especially for a losing position.

2.3 Exchanging Tiles vs. Placing Tiles

There is a discussion by Arneson in 2020 ([2]) on when a player should opt to exchange tiles as opposed to placing tiles on the board. It states that "It is usually a better move to make any possible play than to exchange tiles". This seems to corroborate the assumption in the research by Romero et al. (2009, [4]) which does not consider exchange moves explicitly separately from non-exchange moves. It also mentions that in cases where making any move would open up the high multiplier spaces on the board, it might be favourable to exchange or pass. This is accounted for by the defensiveness score given to each move by Romero et al.

Quackle however does not completely agree with this methodology. The Championship player in Quackle gives extra equity to an exchange move which is played before any tiles are placed on the board. This is probably leveraging a slight starter's advantage that play might have to improve their opening play and make it more likely to be a bingo, due to increased freedom. It could also be because from the starting position, fewer multiplier spaces are reachable. As for a defensiveness rating, there is nothing explicit given in Quackle.

Why is the discussion about how different AIs work important? Well, it is important as it helps us to find out how a human player can think about the strategies which an AI take from a human perspective, and take advantage of that knowledge. It can give a more qualitative way to give hints to the player.

Chapter 3

Ethical issues

As the aim of this project is to eventually help people become better at playing Scrabble, there might be a need for user feedback, evaluation and testing. This would most easily be done through surveys and asking users to try interacting with the game in person.

The responses to such a survey would have to be recorded, which has potential GDPR issues. The most sensitive information in such a survey might be the user's age, which would be used to guide what different age groups might want to see in such a product. In such a situation, an aggregation of the age groups would be enough to remove almost any way to identify the person taking the survey and their response.

A very unlikely, however still possible, alternate use to this project is in decrypting messages from partial text. The reason this is unlikely is that with the encryption protocols which are used today, you have either decrypted the whole message or none of it. In the past, a message could have been partly decrypted, in which case a person might try to guess what the unencrypted letters of a message mean to help them figure out the rest of the cyphertext. A trivial example would be if a simple substitution cypher is used, and the person decrypting the text is looking at the word 'asendpte', where they know that they got the letters 'e', 'd', and 't' correct (e.g. due to the other words in the text). Let's say that the intended word is 'anecdote'. The person decrypting this could use something like Quackle or MAVEN by having a board layout such as 3.1 to see what word suggestions come out, and then systematically try out the ones which correspond to the row which contains the letters 'e', 'd', 't', 'e' to see if the rest of the text makes sense with the substitutions of 'a' $\rightarrow$ 'a', 's' $\rightarrow$ 'n', 'n' $\rightarrow$ 'c', 'p' $\rightarrow$ 'o'.

Both of the above-mentioned considerations are highly unlikely to be issues in the real world. However, what is most likely to be an issue is that there might be a possible copyright issue with the name Scrabble, its board layout, and the lexicon of words used for its gameplay. There does not seem to be an official list of words one can easily download and make use of. This could potentially limit how well the AI does in forming words, along with the number of new words a player learns from playing a game of Scrabble. These issues could most easily be fixed by: using a custom lexicon which is not exactly the one used in official Scrabble tournaments; using a different board layout (similarly to what Zynga's 'Words With Friends' game has done), and potentially differently scored tiles. However, solving the issue in such a way would not be able to as directly address the issue of 'how to become a better player at Scrabble'. Since this is just an undergraduate project with no intention of creating a publishable product with monetization, copyright issues such as these are not a big issue.

Another copyright issue might be the reuse of code, such as that of Quackle. However, Quackle is licensed under the GNU GENERAL PUBLIC LICENSE v3[1], meaning that as long as the license itself is kept in-tact, anyone is allowed to make us of the code as they wish. The intention has been to take full advantage of this.

[1] Quackle license on GitHub

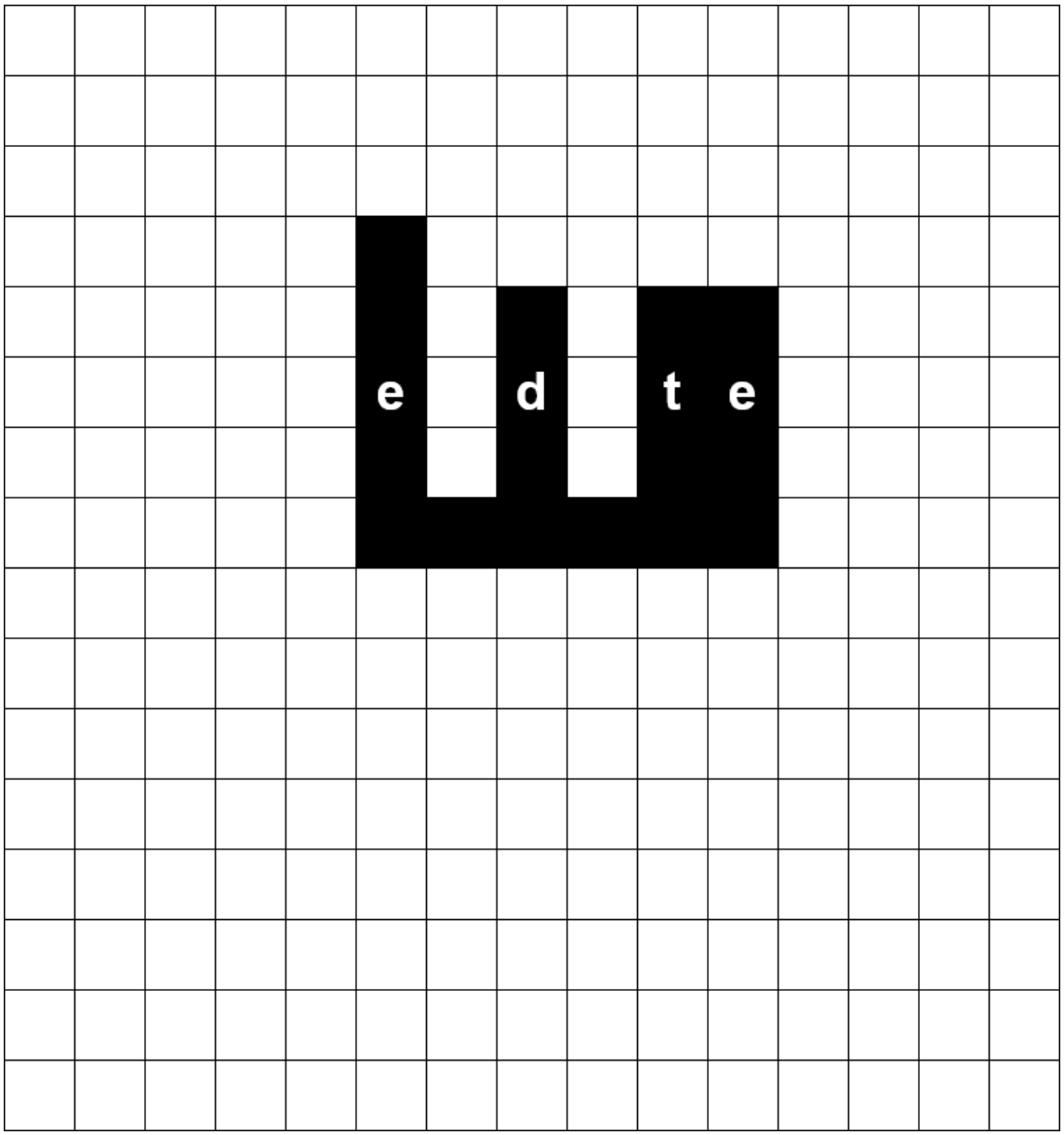

Figure 3.1: Board layout used to help in the decryption of simple substitution cypher. The player's rack size is potentially increased to contain multiple copies of every letter. One of the suggested words to place down could be 'anecdote' which would be the correct decryption of the word 'asendpte'.

Chapter 4

Technical Details

4.1 Requirements

The developed software should be able to show hints to the player. These hints should be understandable by a human, and through multiple games, help them to improve at playing the game faster than without them. Eventually, they should be able to 'generate' these hints for themselves and reason why they made a certain move similarly to how these hints would. There is a basic assumption that the user has a developed lexicon, which this software does not focus on teaching. Other better tools (such as Wordle[1] or jklm's BombParty[2]) for expanding one's own lexicon can be used in a simpler environment. However as we will see later on, what has been developed does manage to somewhat help expand the user's lexicon anyway.

The moves which a hint might suggest should be of high quality. This means that moves generated should, first of all, create valid words according to the lexicon being used, but more importantly, be high-scoring in the long term. Hence, the generated hints need to be strategic, and not only think about the short-term score improvement.

It is not necessary for the hints to be generated quickly. Favour is given to more easily understandable/intuitive hints over the speed at which they are generated. The user should be able to 'decypher' what the hints mean without too much effort. However, hint generation should also not be so slow as to make the player give up using the software. There must be a balance.

This will be built on top of Quackle ([7]) so that less time can be spent developing a Scrabble game. It also means that the hints generator can be built upon existing, shown to be high-calibre, AI - Quackle's Championship player. This almost guarantees that the moves which the hints suggest are going to be good moves.

4.2 Design

Since the software is built upon the existing Quackle code which is mostly in C++ (4.1), the hints generation was also mainly done in C++. The code was compiled using makefiles generated by qmake[3], and automation/aggregation of simulations was done in python.

The existing code made use of the Qt[4] framework, mainly for the user interface (UI). This is also a C++ library, so using it was not too challenging.

[1] https://www.nytimes.com/games/wordle/index.html
[2] https://jklm.fun/
[3] https://en.wikipedia.org/wiki/Qmake
[4] https://wiki.qt.io/About_Qt

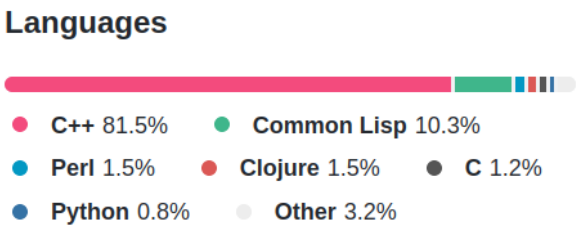

Figure 4.1: Languages used according to GitHub.

Before any additions, the Quackle program looked like figure 4.2. After the main additions, it now looks similar to figure 4.3. Apart from the visual changes, the Greedy AI was also added to the AI arsenal of Quackle for a greater diversity of AI opponents (and hint generators) to choose from. Quackle also provided functionality for two human players to play against each other like in a normal Scrabble game and has checks to make sure that plays are made within the rules of Scrabble.

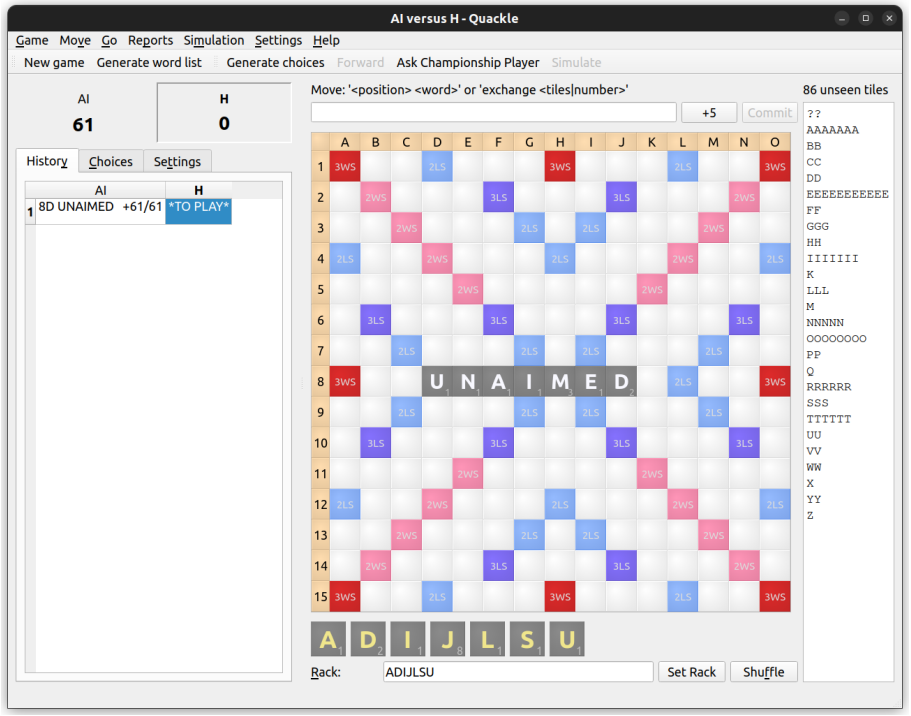

Figure 4.2: Default look of Quackle

What these changes mainly entailed was the addition of hint generation. There were already pre-existing AIs such as Championship player and Static which were altered to be useful in hint generation. Each move was also changed to record the reasons behind all the calculations which were carried out to determine its ranking within an AI's moves list. This is where most of the textual strings come from in the bottom-right hints section.

A big class which required implementation from scratch was 'HintsGenerator'. This is the class which allows for the hints of all AIs to be collected together. It makes use of the newly implemented 'Hint' class which allows for easier collection of the hint messages from the move calculations carried out. After this, 'HintsDisplay' which relies on 'HintsGenerator' displays these hints.

4.3 Implementation

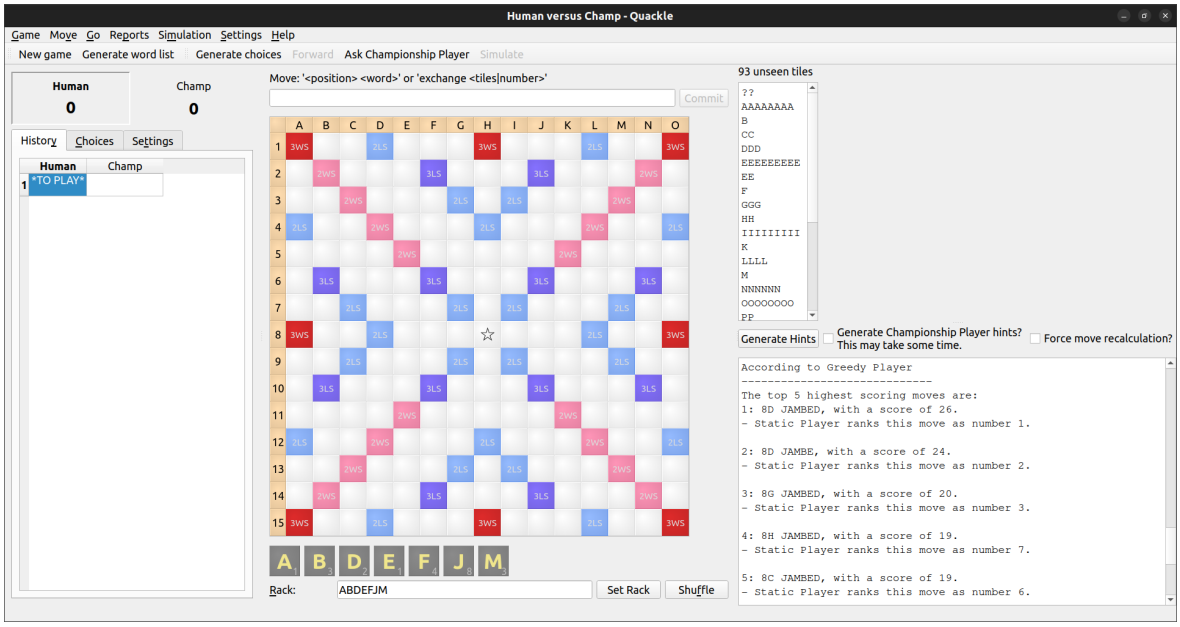

Figure 4.3: What the game looks like overall after the hints alterations.

The decision made was to use Static as the main AI which is built upon for hints generation. There are multiple advantages to this. However, before we can fully appreciate these, we have to be able to understand how the various AIs work.
Three AIs are used for hint generation in this program: Greedy, Static and Championship. There was insufficient time to try to implement other AIs such as MAVEN (Sheppard, 2002, [14]) or 'Heuri' (Romero et al, 2009, [4]).

All of the AIs firstly start by generating all possible words on the board ('kibitz' all words) using the method explained by Gordon (1994, [5]).

The simplest of the three AIs is Greedy. After all words have been generated, it sorts all of the possible moves based on score, and plays the highest-scoring move. If there are no word possibilities, it generates a move that exchanges as many of its tiles as possible (this depends on how many tiles are left in the bag and can range from 0 to 7 exchanged tiles). There are two flaws with this: the first one is that it is disadvantageous to exchange a blank tile, and the second is that this only achieves a locally maximum score. It does not think ahead or consider how the rack leave could affect the next move. The next move will likely be disadvantaged dues to most 'good' tiles already having been used up. Using this AI, the hints which can be presented to the player are limited to just showing them a certain number of top moves. The default number of top moves chosen was 5, as it is unlikely that a user would be interested in reading (or learning) any more than 5 potentially new words per turn. It also means that all of them fit on the screen without scrolling (as seen in 4.4).

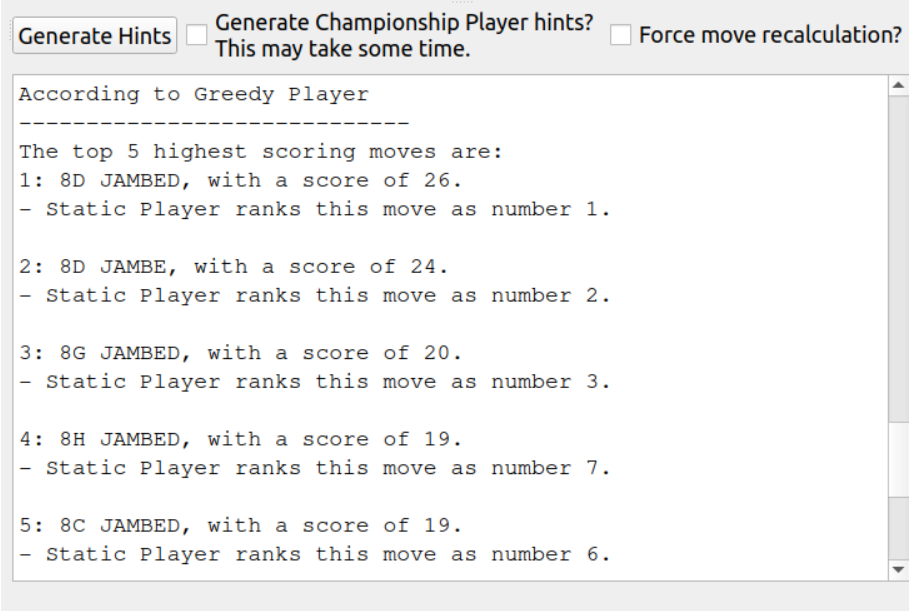

Figure 4.4: The hints generated using the Greedy AI for an empty board with rack 'ABDEFJM'. This is in the bottom right section of 4.3. Scrolling up and down would allow us to see the hints generated by other AIs (Static/Championship).

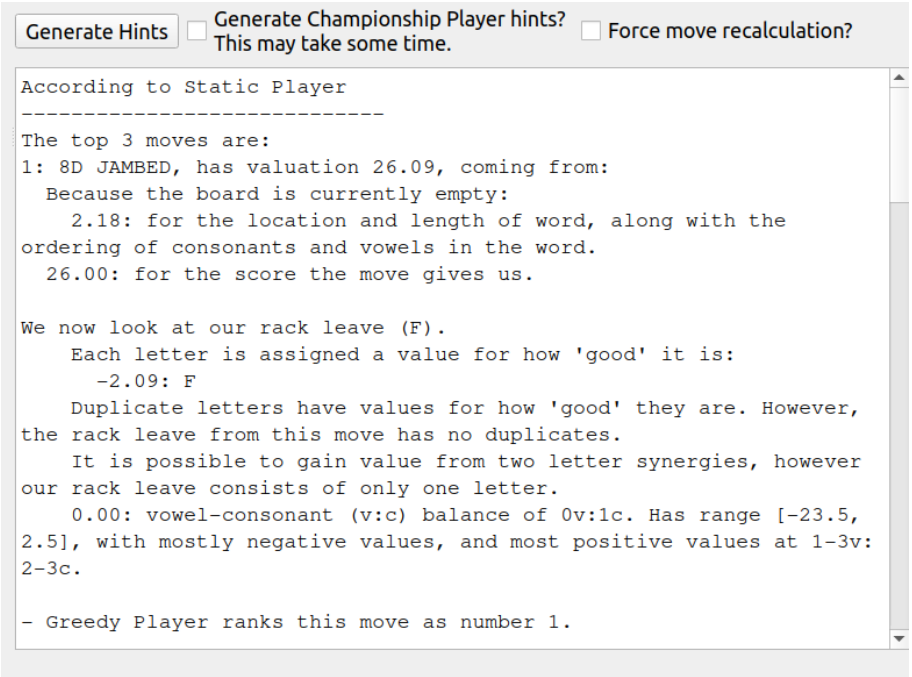

Figure 4.5: The hints generated using Static for an empty board with rack 'ABDEFJM'. Rack leave's equity calculation is further broken down into equity for individual letters, pairs of letters, and consonant-vowel balance.

The next step up is Static. It works very differently from Greedy. After the kibitz stage, it also generates all possible exchange moves (so for a bag of at least 7 tiles, there would be at most $\sum_{i=1}^{7} C(7,i) = 127$ of these moves). Then, each move gets an equity score, which is a heuristic score for how good the move is. If there are no moves possible, then a pass move is played. After this, the moves get sorted by equity, and the one with the highest equity is played.

Except for a pass move, the equity cannot be below −40. For all other moves (exchanges or word placements), it is procedurally calculated and depends on multiple factors: the score of the move, the rack leave's equity, the position of vowels and consonants in the word played, the length of the word, whether this move is made on an empty board, the number of tiles left in the bag, and in the end-game, the tiles the opponent has. We can see an example Static hint in 4.5.

Rack leave is also considered in exchange moves. Note that it doesn't statistically analyse the tiles in the bag and add equity for what we might draw from the bag. It ignores that completely and calculates the rack leave's equity as if the exchanged tiles were played.

An interesting observation with Static is that a bingo is almost always played due to how much its score adds to a move's equity. We can see an example of this in 4.6. It also has no rack leave to consider. So having a large lexicon does affect how well you can play, as the opportunity to play a bingo on average opens up just over twice in a high level game[5] [6] using Static's strategy.

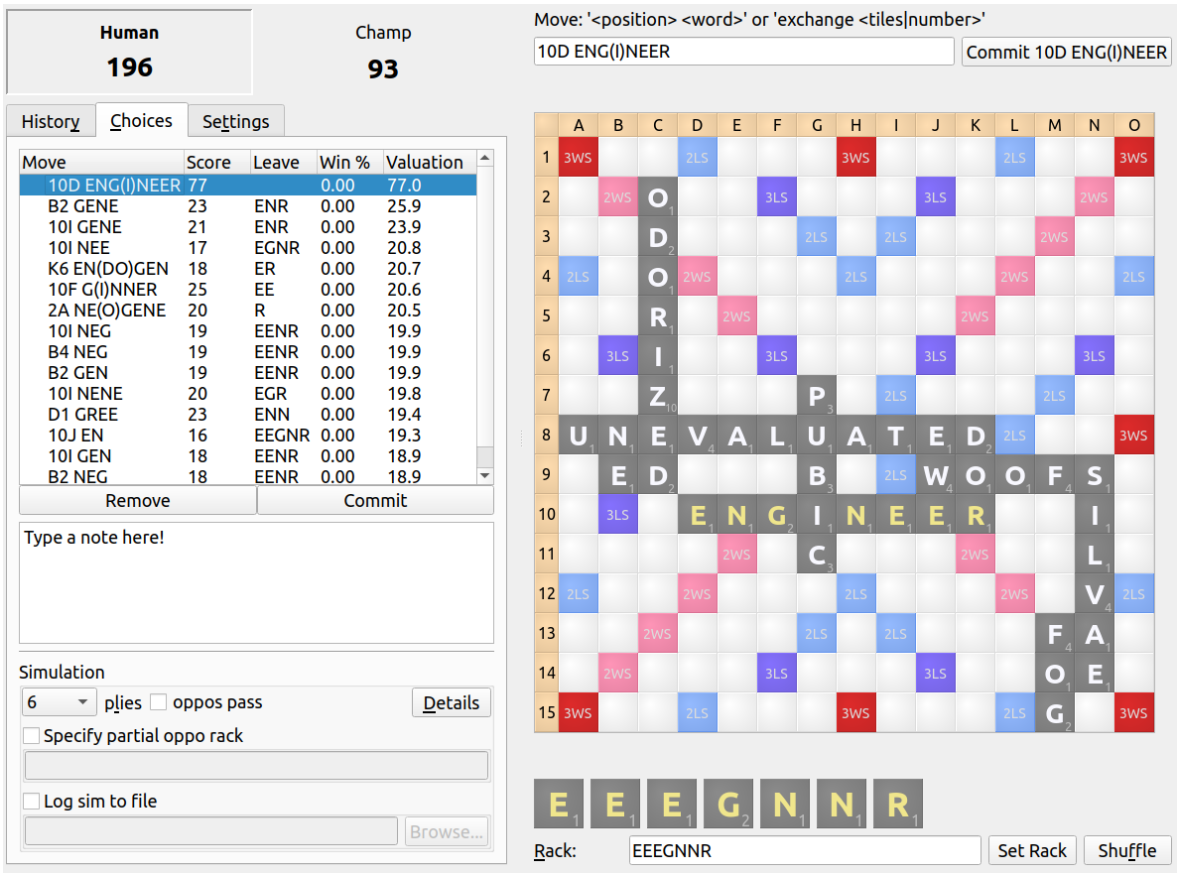

Figure 4.6: Static ranks a bingo as number 1 almost always. On the left are kibitzed moves sorted by equity, with the board position on the right. We can see how large the gap in equity (valuation) is between the top two moves. The hint given for this is 4.7.

[5] https://github.com/michael1212-creator/quackle/blob/simulations/quacker/Static1_v_Static2_results/results

[6] https://github.com/michael1212-creator/quackle/blob/simulations/quacker/Static_v_Champ_results/results

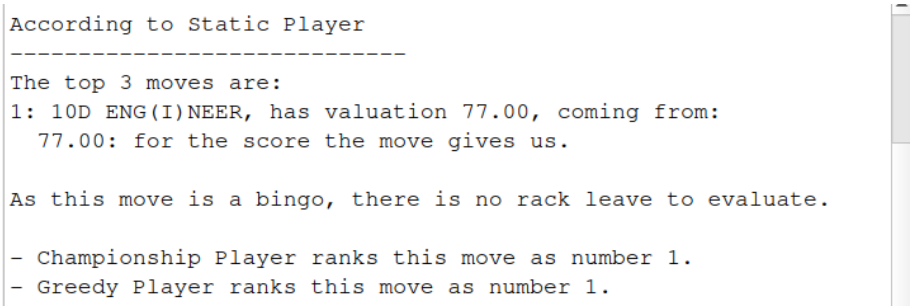

Figure 4.7: Static's hint for a bingo for position given in 4.6.

Finally, there is Championship. This AI builds upon Static. It first takes some number of the top equity-rated moves from Static, filters out ones which are too low under the top scorer (so for example in a bingo), and simulates (hundreds of) future moves of the opponent and itself after that move is made. These simulations are done by playing the move which is being considered and then playing the follow-up moves according to Static. Win% is determined by the percentage of simulations which end up in Championship having a higher score than its opponent. This means that the hints that Static gives are part of the hints that Championship would also generate. However, the simulation part is difficult to make hints for and was hence not explored further. In the current state, Championship hints look the same as Static hints, apart from rankings usually differing.

Once the player commits a move they want to make, there is a quick summary given by all AIs which were tasked to generate hints, as seen in 4.8. This is the only place where Championship 'hints' add extra value.

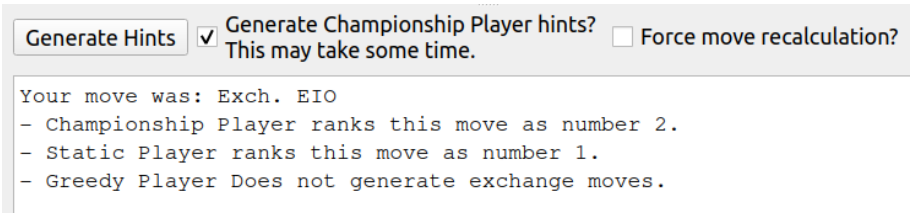

Figure 4.8: Hints given after player commits a move. Shows what each AI ranked its move as.

Another reason to prefer Static over Championship for hint generation is for its speed. Static generates hints almost instantly, whereas Championship can take upwards of 30 seconds to generate its hints. This is due to the time it takes to run its simulations. This is also the reason why there is a 'Generate Championship Player hints?' tickbox above the hints display, as can be seen in 4.8 for example. As we will see in the evaluation (5), Championship is better than Static, however in terms of Elo rating, they are very close, meaning both would count as professional-calibre Scrabble players.

Greedy also generates hints almost instantly, and even slightly faster than Static, however as we will see in the evaluation (5), it doesn't perform as well as Static in gameplay.

4.4 Enhancements

To be able to generate humanly understandable, intuitive hints, any original code which does any calculation related to how good a move is had to be altered. These alterations usually involved adding more code which generates text based on the calculation. This would have an overall effect of slowing down the execution of the program, and hence make it less efficient.

For example, in the equity calculation for the rack leave, the original code had two options: find the equity using a lookup table, or calculate it on the fly. The lookup table method had to be removed in favour of the procedural generation, as it allowed for a more detailed hint message. With only the large

lookup table, all that could be presented to the user would be the equity for their rack leave, without any explanations of single letter or letter pair equities. This does have the desired effect of reducing the amount of memory required to run the program by about 8MB, however, this is negligible compared to 8GB.

This is not a noticeable increase in time for Static, however since Championship simulates so many moves using Static, this inefficiency adds up to make it slower.

Any move which is generated, whether played or not by an AI, has its score calculated. This makes generating hints from Greedy an easy task: after generating all possible moves, find the top 5 moves, and display the moves along with their score.

The process is longer when considering moves generated by Static. This is firstly because Static works on equity and not score, and secondly, because Static also generates exchange moves. So, each move is fed through a 'catchall evaluator' (or just 'evaluator'). This evaluator calculates its equity as explained in the above 'Implementation' (4.3) section. Each time an equity calculation is carried out (and on some occasions even when not carried out, such as the 'bonus' or when rack leave consists of a single letter - 4.9), a message is generated associated with it.

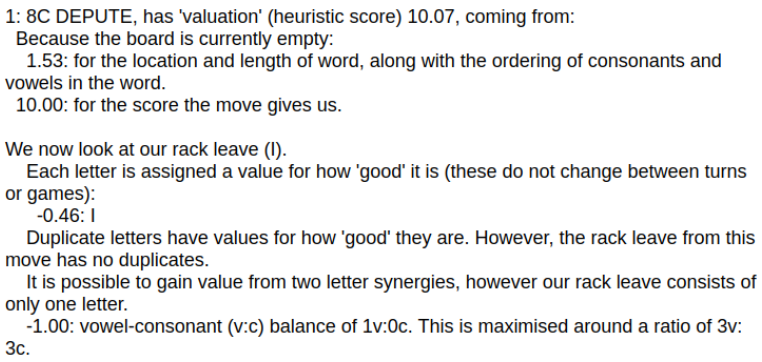

Figure 4.9: Even though no equity calculation is carried out, there is a hint message (near the bottom) telling the user where they could have possibly gained equity from. Total equity $= 10.07 = 1.53 + 10 - 0.46 - 1$

As Championship builds upon Static, it retains these hints and allows for future message additions in the 'Hint' class.

Chapter 5

Evaluation

There does not appear to be a current state-of-the-art in the field of helping Scrabble players become better. So, the bar is set very low. Any kind of useful hint which helps the player learn is better than what there currently is - nothing.

The most basic goal of this project has been met. The program developed does indeed generate hints. And since these hints come from AIs which seem to be able to play Scrabble at a decently high level as we will see below in the 'AI Competitiveness' section (5.1), the hope is that the quality of hints is high, and so should help users to get better at playing Scrabble.

Two aspects of hint generation require evaluation: how good the AIs generating the hints are, and the quality of the hints which they generate. The former has to be evaluated so that we know that the hints which it generates are competitive, since the hints are based on the top ranking moves which the AIs make. Evaluation of the quality of hints is required as this is the part which actually teaches the user to play better.

5.1 AI Competitiveness

The first side of the evaluation is evaluating the AI itself. If the underlying AI which generates the moves is flawed and is a bad Scrabble player, the hints it produces would also be bad. Hence, we also need some way to evaluate how good the moves the AIs generate are. These types of measurements can be done quantitatively in contrast to the qualitative nature of hints.

Appel and Jacobson (1988, [1]) Gordon (1994, [5]) suggest that the success of an algorithm can be measured and by how many words the lexicon being used to generate the moves contains (the more the better), and how long the move generation itself takes (the less time the better). Another metric used in Appel and Jacobson's work is how much memory the lexicon's representation takes up. As this paper is quite old, memory capacities were small in 1994 (and even smaller in 1988), so lexicon size (and its representation) was a real consideration. Today, this is not as important - most PCs have at least 8GB of memory[1]. Even a semi-minimized, expanded (uncompressed) GADDAG representation of the lexicon is only 27MB. This is insignificant compared to 8GB, meaning that as long as there is a sensible implementation of the data structure, we are not limited by the amount of memory a system has.

The whole game takes up under 200MB in memory. This includes the lexicon which is used and all of the possible AIs which can be chosen as players. This is an insignificant amount of memory for modern-day systems, and so can be counted as a success.

Hence, when it comes to evaluating the AIs, the best way to evaluate them is through how quickly they can make a play or help to generate the learning hint - no one wants to wait a long time to be given a hint. This is one of the easier things to test as it does not rely on other people using the software -

[1] according to the Steam Hardware Survey of December 2021: https://store.steampowered.com/hwsurvey, and since Scrabble is a game, we expect this to be representative of the type of system a person wanting to use this software would have.

simulations of various scenarios can be done (such as no wildcard, one wildcard, or two wildcards, all with varying board and rack layouts) to determine an average, worst- and best-case scenarios. As long as the testing of how long it takes for a task is done on the same system, with the approximately same number of background tasks, the results should be reliable.

Under the lexicon size criteria, all three AIs which were investigated made use of the entire lexicon with acceptable words. This means that under this criterion, the AIs are perfect. In terms of speed, Greedy and Static generate moves instantly, which once again makes them perfect. Championship however is not perfect in this regard, as in many cases it can take over 20s-30s for it to make a move.

An easier (although less reliable) way to measure the AI's goodness is to have them play against each other multiple times and measure statistics such as the distributions of scores per game, win percentages, number of bingos per game and mean score per turn per game. This can then be compared to the distribution of other AI vs. AI games to perhaps determine a 'best' AI, and tell us if any of them are even suitable for hint generation.

There are also some clearer ways to evaluate how good the AIs are, such as having them play against real people of known Elo rank. This can allow us to establish an Elo rank of the AI and clearly say how good it is in the exact same way that actual Scrabble players are ranked. This would require being able to deploy this online and get a large enough user base, which is out of the scope of this project. Online published Elo ratings will have to suffice.

Static AI wins 40%[2] of matches against the championship player, implying that their Elo ranking is within 100 points ([10]). This result is from 1022 simulated games of the Static AI vs. the Championship AI, where Static wins 416 of the games, and draws 1. Sheppard's MAVEN paper (2002, [14]) in section "10.1. Skill metrics" states that a good measure of an AI's success is its mean score per turn, comparing that of static which is 35.25 to that of Champion which is 35.91 shows us that Championship is only a small improvement. This seems to suggest that Static is a good Scrabble player.

The highest ranked Scrabble player Adam Logan has a peak Elo of 2188 ([8]). Quackle's is 2232 ([10]). Since Static is about 75 points below Quackle, this means that Static's Elo is 2157. This is very close to Logan's - only 31 Elo difference - suggesting that Static is a powerful opponent. So basing hints on Static seems to have been a valid choice.

Although Greedy generates hints instantly like Static, it is worse than Static. It only wins 25.0%[3] of games (in a sample size of 1142 games) and has a mean score per turn of 32.9, compared to Championship's 36.3. This is a much wider mean score gap than that of Static and Championship. When matched up against Static, it wins 34.2% [4] of games, with a sample size of 1136 games.
Interestingly enough, however, the theoretical Elo rating of Greedy would be within 200 of Championship's, still placing it with a similar ranking as some of the top Scrabble players.

[2] https://github.com/michael1212-creator/quackle/blob/simulations/quacker/Static_v_Champ_results/results
[3] https://github.com/michael1212-creator/quackle/blob/simulations/quacker/Champ_v_Greedy_results/results
[4] https://github.com/michael1212-creator/quackle/blob/simulations/quacker/Static_v_Greedy_results/results

5.1.1 Summary of findings

Player	MAVEN	Championship AI	Static AI	Greedy AI	Human Expert
Peak Elo rating	1892[5]	2232[6]	N/A	N/A	2188 [7]
Win% vs. Championship based on known Elo rating	19%	50%	N/A	N/A	45%
Relative Elo rating to Championship based on simulations	N/A	±15	-75	-200	N/A
Win% vs. Championship based on simulations	N/A	51.7% / 47.7%[8]	40.7%[9]	25.0%[10]	N/A
Mean time to generate moves	N/A	24s (maximum observed: 70s)[11]	0s	0s	unknown
Mean score per turn when playing against itself	35.0 [12]	34.9[13]	36.3[14]	32.5[15]	33 [16]
Number of bingos per game when playing against itself	1.9 [17]	2.3[18]	2.2[19]	1.2[20]	1.5 [21]

5.2 Quality of Hints

5.2.1 Evaluation Methods

In an ideal world, the best way to evaluate the quality of the hints would be to have two large groups of people, approximately of the same backgrounds and ages. Over the span of a few months, we measure how much the participants of each group have improved their gameplay. This could be done by having the control group play Scrabble without the learning mechanism and the other group with the option of the learning mechanism. Then, at the end of each week, every participant is asked to play and submit three unaided games against the Quackle AI. From this, we measure the difference in scores between the player and the Quackle AI and divide it by the sum of the two. We can plot this as a function of time to see the improvement of each player over time, hoping that the players with the learning mechanism

[5] As seen in [9]
[6] As seen in [10]
[7] As seen in [8]
[8] https://github.com/michael1212-creator/quackle/blob/simulations/quacker/Champ1_v_Champ2_results/results
[9] https://github.com/michael1212-creator/quackle/blob/simulations/quacker/Static_v_Champ_results/results
[10] https://github.com/michael1212-creator/quackle/blob/simulations/quacker/Champ_v_Greedy_results/results
[11] Calculated by timing a game of Championship vs. Championship
[12] As seen in section 10.2.1 of Sheppard's 2002 [14]
[13] https://github.com/michael1212-creator/quackle/blob/simulations/quacker/Champ1_v_Champ2_results/results
[14] https://github.com/michael1212-creator/quackle/blob/simulations/quacker/Static1_v_Static2_results/results
[15] https://github.com/michael1212-creator/quackle/blob/simulations/quacker/Greedy1_v_Greedy2_results/results
[16] As seen in section 10.2.1 of Sheppard's 2002 [14]
[17] As seen in section 10.2.1 of Sheppard's 2002 [14]
[18] https://github.com/michael1212-creator/quackle/blob/simulations/quacker/Champ1_v_Champ2_results/results
[19] https://github.com/michael1212-creator/quackle/blob/simulations/quacker/Champ1_v_Champ2_results/results
[20] https://github.com/michael1212-creator/quackle/blob/simulations/quacker/Greedy1_v_Greedy2_results/results
[21] As seen in section 10.2.1 of Sheppard's 2002 [14]

improve faster, potentially with a higher ceiling before plateauing.

Sheppard's MAVEN paper (2002, [14]) in section "10.1. Skill metrics" suggests that the final player score is not very representative of the skill of a player, and so perhaps neither is the ratio method outlined above. It instead suggests using the average number of points scored per turn. This is especially useful when the two opponents are of different calibre, as we expect to be the case, especially when the players are just starting to learn to play Scrabble. It does also state that such a measure is perhaps more useful for measuring the performance of an AI and not a human player, suggesting instead an Elo-rating system for humans. In a large-scale experiment like the one outlined above, this is perhaps doable - every player plays against various other players in the test groups. Then, we would plot the players' Elo ratings over time.

However, as this project has been tight on time, this opportunity never presented itself. So, the best way we would be able to evaluate its success would probably be through qualitative feedback from users. This could be done by distributing a survey, giving the participants a chance to play the game, and evaluate how useful the learning messages given are to them. The survey could have open-ended questions such as what they think of a certain hint if the suggestion is sensible or intuitive, and whether they think the tool helped them develop their play.

Since no easy way of distributing the software to survey participants has been developed, another way would be to distribute various scenarios. What is meant by 'scenario' in this case is a board layout, the current player's rack, the generated hints, along with both of the players' current scores, and the unseen tiles.

Alternatively, it would be good to sit down next to someone using the software and observe how they use it and ask questions while doing so. This is the approach taken here.

5.2.2 Feedback From Users

Four potential users were interviewed for 60 to 90 minutes. They will be referred to as P, Z, T and H. Z and T were in-person, whereas P and H were over a video call with screen sharing and control sharing. They were interviewed in the order as their names are listed, and any small changes which they suggested were made before the next was interviewed. Note that pre-existing features of Quackle are not being evaluated, so comments aimed at ease of use of pre-existing features are disregarded.

Firstly, they were allowed some time to get used to the software and how it works. After this, their attention was directed towards the bottom-right of the program - the hints generator. They were asked for comments on the locations of widgets. Then, a two-player game against an AI of their choice was started, with the question of 'why did you choose that AI?'. Once in the game, they were asked to comment on their experience with accessing the hints which can be generated. Finally, they were asked about how good or useful they thought the hints are, whether they are likely to use use them, and their general thoughts on improving their Scrabble gameplay if that was a goal of theirs. Attempts were made to also get the user to comment on how long hint generation takes - such as by asking how long they would be willing to spend on a game, but without a direct reference to the time it takes to generate hints, as to not skew their opinion.

P's Feedback

P's primary language is also not English. They also have no experience with Scrabble or its rules. They were using a laptop with a small 14" screen, which also helped identify less visible elements of the game. After an explanation of the rules of the game along with a short partial game to show what that practically looks like, a new game was started.

One of the first things noticed was that it was not very obvious that elements on the screen can be resized. This is a display limitation of Qt which can be difficult to address sadly. A small screen also does not help. They chose the 'Twenty Second Championship Player' (which is the Championship player AI but limited to a maximum of 20 seconds of simulation time) with the reason given being that they didn't want the AI to be too hard. There were no major concerns about how long the game would take. They said that on second thought, they might have even wanted to choose an easier AI. There was an

interesting comment made about balancing the layout between the bag and hints.

They were not impressed with having to resize balance the size of the bag and hints each time they wanted to see one or the other. Their preference would have been to have the bag at the very top, spread out horizontally as opposed to vertically, which would allow a lot more free space for the hints below it. It was. They also had no idea what the 'Force move recalculation' tickbox might do.

"No-one reads help" was another comment made, after directing their attention to the 'Help' menu. This was accompanied by the two comments of it only happens when users are "desperate", as otherwise "they would just play and see how it works". This is a valid point, and hence why the help option does not contain critical information. It (5.1) only contains details about the inner workings of the various hint-generating AIs. The aim has been to have an intuitive interface that explains each element on its own.

An expected comment was about the organisation of hints. Navigating through the various AIs' hints is not easy. Their suggestion was to make some words bigger or bold just to distinguish them. My suggestion of putting the various AI's hints in different tabs "is even better".

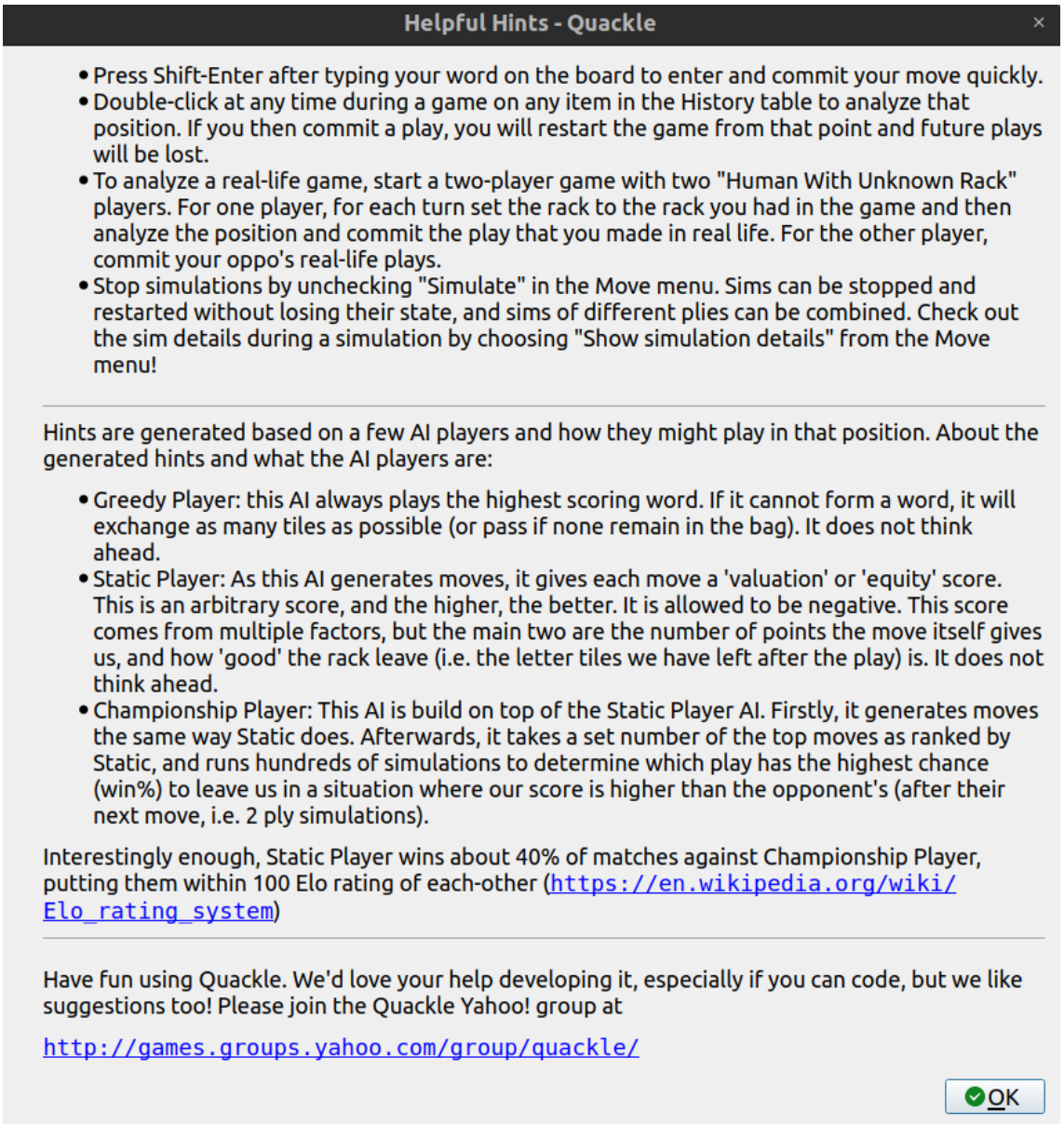

Figure 5.1: The game's help menu.

P's main concern in improving their scrabble play is their limited lexicon, and hence a comment they made was "I think they [the rate of improvement when comparing using and not using hints] would be the same" since hints wouldn't help with lexicon extension. Diving deeper into this, it turned out that perhaps it's not a limitation of the lexicon, but a limitation of solving anagram problems. This is an implicit limitation of the assumptions made when developing this software, however just like with extending the user's lexicon, hopefully, the more games played using the hints generator, the more the user's anagram solving ability develops.

When talking about strategy, it turned out that heuristics are not something P is familiar with. Not being able to gauge a score from equity was something that P seemed very disappointed with. This is more likely an issue of evaluation method than difficulty with the program since if it was possible to give a user time with the program to use it for a longer time, they would become familiar with the equity system and have an intuitive understanding of its scoring. However, this takes time which is not available in a short interview. There was still some useful information that was gathered. To P it was not obvious that the equity which a letter or pair of letters in the rack leave award is constant between games. We can see these hints in 5.2. Another comment was made about how it was unclear what a hint is trying to say about the range of a contribution, as seen in 5.3. It was suggested to just remove the mention of ranges.

```
We now look at our rack leave (CT).
    Each letter is assigned a value for how 'good' it is:
      0.74: C
      -0.20: T
    Duplicate letters have values for how 'good' they are.
However, the rack leave from this move has no duplicates.
    Two letter synergies:
      0.34: CT
```

Figure 5.2: Hints explaining rack leave letter equity before P's feedback

```
    -1.00: vowel-consonant (v:c) balance of 0v:2c. Has range
[-23.5, 2.5], with mostly negative values, and most positive
values at 1-3v:2-3c.
```

Figure 5.3: Vowel-consonant ration range - confusing message according to P

Based on P's feedback the changes made were to add text explaining what 'Force move recalculation' does (5.4); details about ranges of possible values were removed from hints (5.5); it was made obvious that the equity which is awarded for letters in a rack leave is constant between games (5.6).

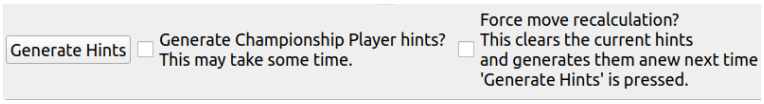

Figure 5.4: Explanation for 'Force Move Recalculation' tickbox

```
    -1.00: vowel-consonant (v:c) balance of 0v:2c. This is maximised
around a ratio of 3v:3c.
```

Figure 5.5: Vowel-consonant ration range removed from hint

```
    Each letter is assigned a value for how 'good' it is (this does not
change between turns or games):
```

Figure 5.6: Hints explaining rack leave letter equity after P's feedback

Z's Feedback

Z knows the basic rules of Scrabble, however, has not played a lot of games. Similarly to T, English is not their primary language. After a short familiarisation with the software and no interaction with the hints generator, feedback questions were posed to them.

Their choice for the opponent AI was Greedy. This was because it is the easiest of the AIs, and they were not very confident with their play. There was no initial concern about the time it would take for the opponent to make a move.

After the game started, their attention was diverted toward the bottom right of the program. The first comment was about there being a bit too much text, especially for the last 'Force move recalculation' tickbox. However, this was not too bothersome, as to them it meant that there is a more in-depth explanation that can be read. I suggested a way in which this can be altered to perhaps decrease the amount of text - a setting that can be toggled making text less verbose. This would assume that the user has used to program before and is familiar with it, so some explanation text could be removed. Z said that that would be an acceptable solution.

It was not obvious that elements could be resized, just as with P. Sadly, there were also no suggestions on how to make it more obvious. Qt allows for colours of elements to be changed, however, this makes the UI not look very good. But since it makes it more usable, the decision was made to change the colour of the splitters.

There was also a comment made about the font used. The monospace font 'Courier' looked too aggressive, too digital, and not friendly for reading. It was also described as being too "strict" or "serious", especially since this is a game, not some technical writing.

Z was correctly able to infer what the 'Generate Championship Player hints' tickbox does, however, did not fully understand what the other tickbox did. Once it was explained, their suggestion was to perhaps change the word 'move' to 'hint' to make it more obvious. It was also suggested that the 'Generate Championship Player hints' tickbox also mentioned that it only has effect once the 'Generate Hints' button is pressed.

When asked about how much time they are willing to spend on learning Scrabble, they said that they are willing to spend 2 hours per week. Because of this time limitation, they would not always want to want to wait for Championship's moves to be generated. They would only be willing to spend 20-30 minutes with Championship's hints. This is because they "don't have the patience to wait".

Finally, the focus came to evaluating the hints themselves. Z could not find what 'rack leave' meant very quickly. They suggested that it is enough to change the letters in the 'help' menu to be capital or bold. The next question posed was whether they think the hints would be helpful in learning Scrabble when comparing this program to playing Scrabble with no hint generation, and then also compared to a program which only generates the moves, but with no explanation. Z's response was that they would definitely learn faster with full hints, as they explain what makes one move better than another. After multiple games, they would then learn where a move's equity comes from, and be able to reason why one move is probably better than another.

Changes made based on Z's feedback were: changing the word 'move' to 'hint' next to the tickbox explanation for 'Force move recalculation' (5.7); text was added to explain that the 'Generate Championship Player hints' tickbox only has effect after the 'Generate Hints' button is pressed (5.8); the colour of splitters (resize 'bars') was changed to stand out (5.9); changing the font used for hints to 'Arial' (5.10); the letters for 'rack leave' in the help menu were made bold (5.11);

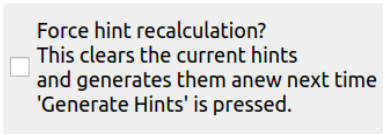

Figure 5.7: 'move' changed to 'hint' for 'Force move recalculation' tickbox

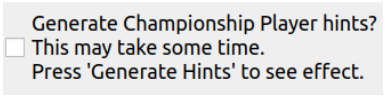

Figure 5.8: 'Generate Championship Player hints' tickbox now says that 'Generate Hints' has to be pressed for it to have effect

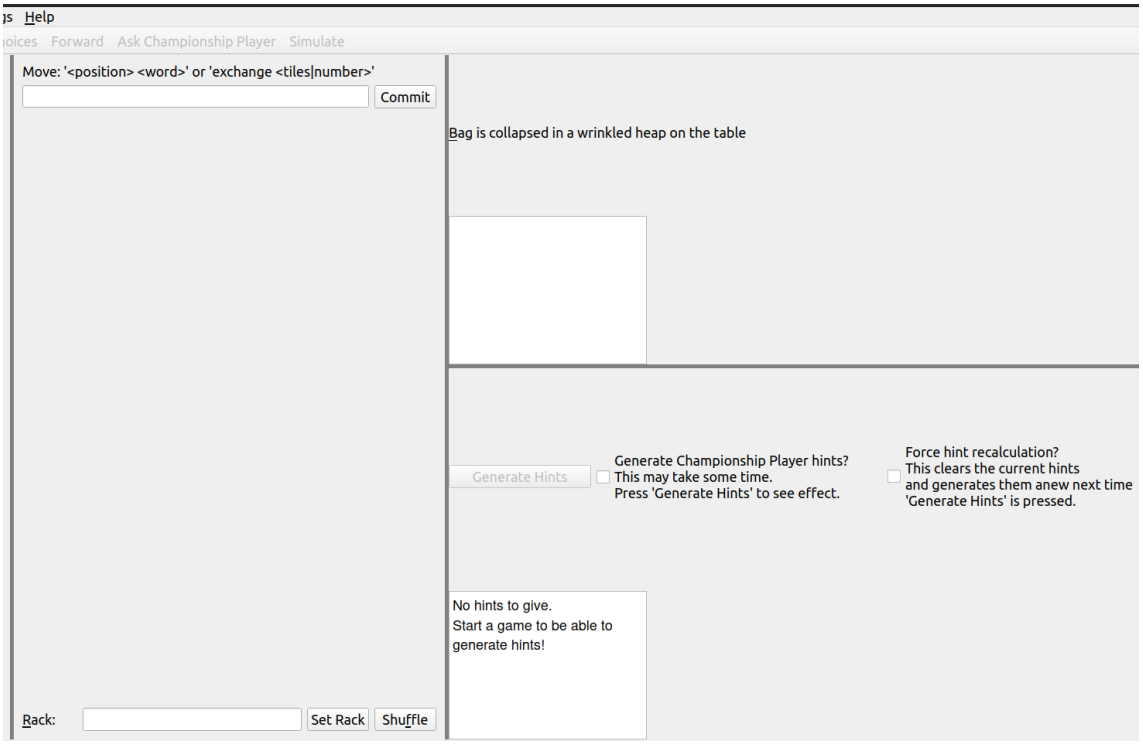

Figure 5.9: Splitters were changed to have the grey colour

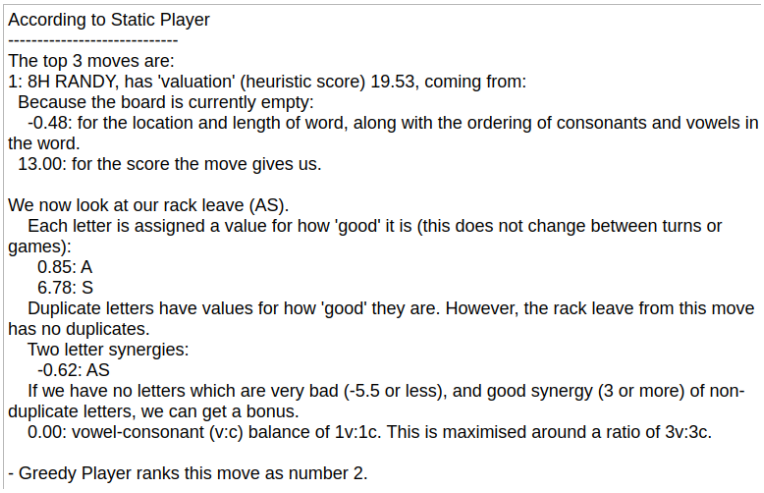

Figure 5.10: Font used for displaying hints changed from monospace Courier to Arial

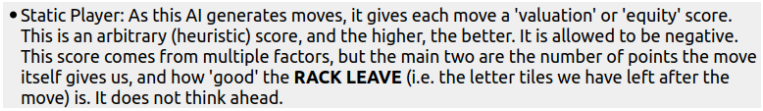

Figure 5.11: 'rack leave' made bold and capitalised in the help menu

H's Feedback

H is a particularly interesting case since they are an avid word games player, so their feedback can be particularly useful.

In their familiarisation round, they played against the Speedy player, which seems to be similar to the Static player. When going into the round with hints, they chose to play against Greedy. The reasoning given seemed to suggest that they wanted to see how various AIs play styles and see if any of them play defensively. Defensive play would include restricting access to bingos or triple-word score multipliers. Greedy does not take any action to be defensive, however Static and Championship can indirectly play defensively near the end-game. This did not factor into H's game, however, it was good to know what players might be looking for.

It still did not seem obvious that the grey bards (i.e. splitters) allowed H to resize various elements on the board. This is concerning, as there is no other obvious way to make the functionality of splitters obvious. "Not until [they] hovered over it" did H realise that it allowed for that functionality.

H seemed happy to learn about the theoretical maximum score that a single play can get in Scrabble. This is generated just after the hints given by Greedy as a comparison data point for players.

Navigating around through the hints given by different AIs was also a problem for H. Addressing this would be a useful improvement to the user experience (UX) of the software for the future. This was further exacerbated by them not having the program right in front of them and using it via remote desktop application, which adds latency.

When reading through the hints, H wondered "why is 'TT' so bad" as a duplicate letter (5.12). They seemed to think that a rack leave with two of the letter 'T' tiles would be a good one as "it is pretty common". This raises an interesting question: could the AI be wrong? Is there a more optimal value that could be assigned to TT? It could also be that the percentage of words in H's lexicon containing two Ts is higher than that allowed by the game's lexicon. It is also possible that what they think does not reflect reality. However, research into creating other Static-like AIs which use the same lexicon would be good to have, as it would allow for comparison of equity ratings such as these. It would be able to make us more certain of the correctness of ratings.

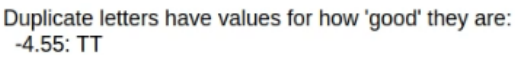

Figure 5.12: Equity awarded for the duplicate letter 'T' is -4.55, which is a bad score

There was also a short discussion about the vowel-consonant ratio score. One of the things asked was "what's the heuristic score of a 3v:3c ratio". The answer was given to them, and when asked whether they would like to have such a number given in text next to the ratio, they said: "I don't think that I would use it in decision making". This seems to support P's feedback that giving the range that an equity rating for something can take is not very useful, and should just be excluded.

They also don't classify consonants and vowels as black and white as the program does. The example they gave was that even if they only had 2 vowels and 5 consonants, but 2 of the consonants were an 'R' or 'L', they would count this as a good rack. This seems to ask the question of if perhaps there is a better way to classify letters than just consonants and vowels. However, the equity of pairs and duplicate letters probably already accounts for this to some extent. H themselves said that "it is good to know which letters work well with which letters", so liked the synergy calculations presented and found them useful.

Explaining what 'Force hint recalculation' does seems to be a difficult task. H was also unable to

tell how it works or what it does. It can be difficult to explain to end-users who are unfamiliar with the concept of caching why having such a tickbox is useful. After an explanation, they were unable to present a way of how it could be improved. What they did suggest was to just "let them figure it out" (them referring to the user). Since this is a game meant to be played multiple times, this is a possibility, however not the friendliest for new users.

A suggestion made by H was about the message shown after clicking commit: it can be improved to also include the evaluation of the play (according to Static) for the word which was played. H would not always play the words which the AI would suggest, and wanted to see a more detailed explanation as to how good their own play was. However, the program did not allow for that. Allowing for the player's move to be evaluated in more detail can be good, as it can allow them to see why it is potentially not as good of a move as they might have thought. Related to this, they also suggested sorting the single letter, duplicate letter, and two letter synergy equities by equity instead of alphabetically. The reason given for this was so that in later games, the player can remember to "do more of this". This comes down to preference. Ideally, there would be a setting allowing you to choose how they get sorted.

H was then asked for their thoughts on the hints - how useful they are, whether they would apply to a real game, and any other comments. The situation suggested was where they've trained using this program, and now they are playing a game against a high-ranking Scrabble player without access to the program. Their first response was about how "it would be a good way to learn new words" and "refresh your vocabulary". In terms of the strategising aspect, their initial thoughts were that it would not be useful to them "as a human". The reasoning was that humans don't need to quantify or score different aspects of the game to reach a decision, we can instead develop an intuition or "vibe" for it. However, they thought that perhaps this is not really possible to do for lower-level players with less experience, and these hints would be useful to them. The reason why it would be useful to lower-level players is that it helps you "learn how to get the vibe. The scoring might be helpful". It also gets them thinking about the different quantities which affect gameplay mentioned in the hints.

H then went on to explain their strategy. The explanation seemed to suggest something similar to Greedy, however sometimes preferring to have 'S', 'ED' or 'ING' in their rack leave, as "they are useful for extension". Developing an AI which plays like this and evaluating its plays against the other AIs might be an interesting research project, however, it is outside the scope of this project. On the other hand, intuition seems to suggest that AIs similar to Greedy would not be better than Static.

H seemed unconcerned about waiting for Championship to generate its hints - the 'Generate Championship player hints' tickbox was ticked throughout the full game they played, even though it was explained to them that the hints given by Championship are not much different compared to those from Static. So there do seem to be users who would be willing to always wait for Championship hints. However, while waiting for Championship hints, they would start reading Static's hints. Since these hints are at a high level as well, there is no concern about learning things the wrong way. It also means that in either case, the hints generated by Static are useful.

There was also an unexpected comment that the win% is useful to them, especially at the end of the game. This is even though there is no explanation as to where that comes from, so it is not as memorable as equity. However, it might help with learning patterns and what might work in an end-game scenario.

The only change made based on H's feedback was to include an in-depth evaluation of the user's play after committing it. (5.13).

Your move was: 8G HE
- Static Player ranks this move as number 12.
- Greedy Player ranks this move as number 210.

8G HE, has 'valuation' (heuristic score) 13.47, coming from:
Because the board is currently empty:
0.22: for the location and length of word, along with the ordering of consonants and vowels in the word.
5.00: for the score the move gives us.

We now look at our rack leave (AEMNR).
Each letter is assigned a value for how 'good' it is (these do not change between turns or games):
0.85: A
1.09: E
0.63: M
-0.16: N
0.51: R
Duplicate letters have values for how 'good' they are. However, the rack leave from this move has no duplicates.
Two letter synergies:
0.67: AE
0.91: AM
0.06: AN
0.05: AR
-0.27: EM
0.02: EN
1.11: ER
0.49: MN
0.47: MR
-0.18: NR
Total synergy = 3.33
If we have no letters which are very bad (-5.5 or less), and good synergy (3 or more) of non-duplicate letters, we can get a bonus.
0.49 = 1.5*(synergy-3): our bonus valuation.
1.50: vowel-consonant (v:c) balance of 2v:3c. This is maximised around a ratio of 3v:3c.

Figure 5.13: The hint given after a commit, after H's feedback. '8G HE' would not have been shown when the 'Generate Hints' button is pressed, since it is not a move ranked in the top 3 of Static. However, this still gives explains the move.

T's Feedback

T is new to Scrabble, but have read up the basic rules for it. English is not their first language. However, they have seen the program in some of its development stages.

T's choice for an AI opponent was Speedy player. The reason for this was because they wanted to play against an opponent which makes moves quickly.

T has a background in web design, so they were able to offer a good amount of advice on how the UI could be improved. The first thing they noticed was that there is a lot of empty space in the top-right (figure 5.14). They suggested changing it and making it look like in figure 5.15 so that there is no empty space and less repetition in the tickbox explanations. This would probably also address P's concern about the wasted empty space and the bag not always being fully visible.

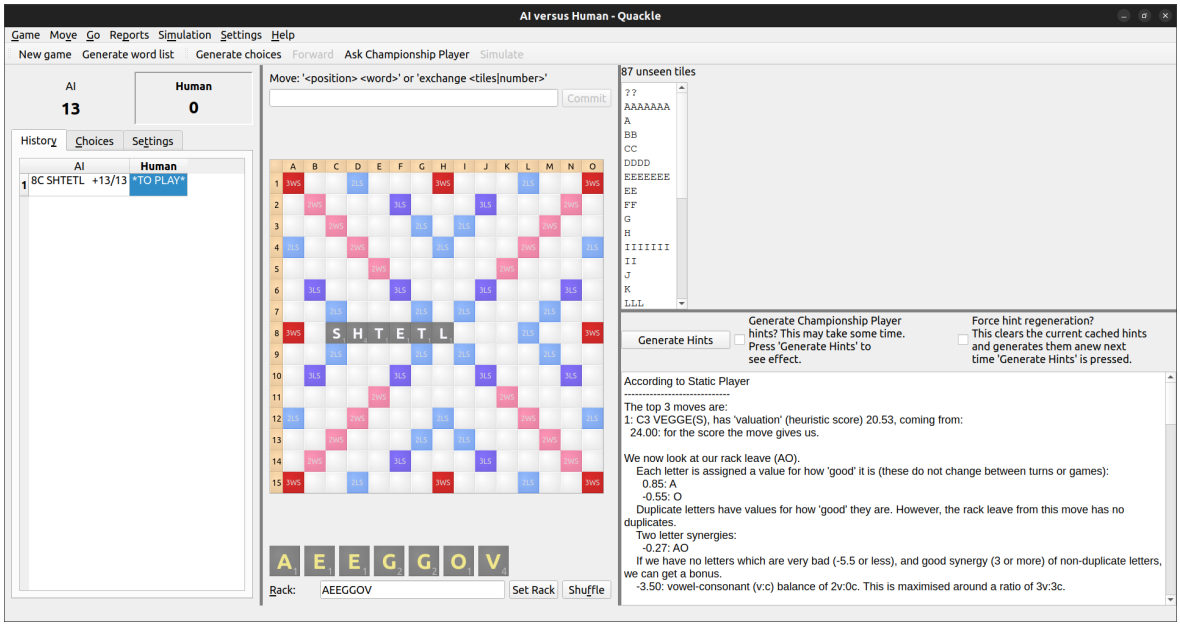

Figure 5.14: There is a lot of empty space in the top-right.

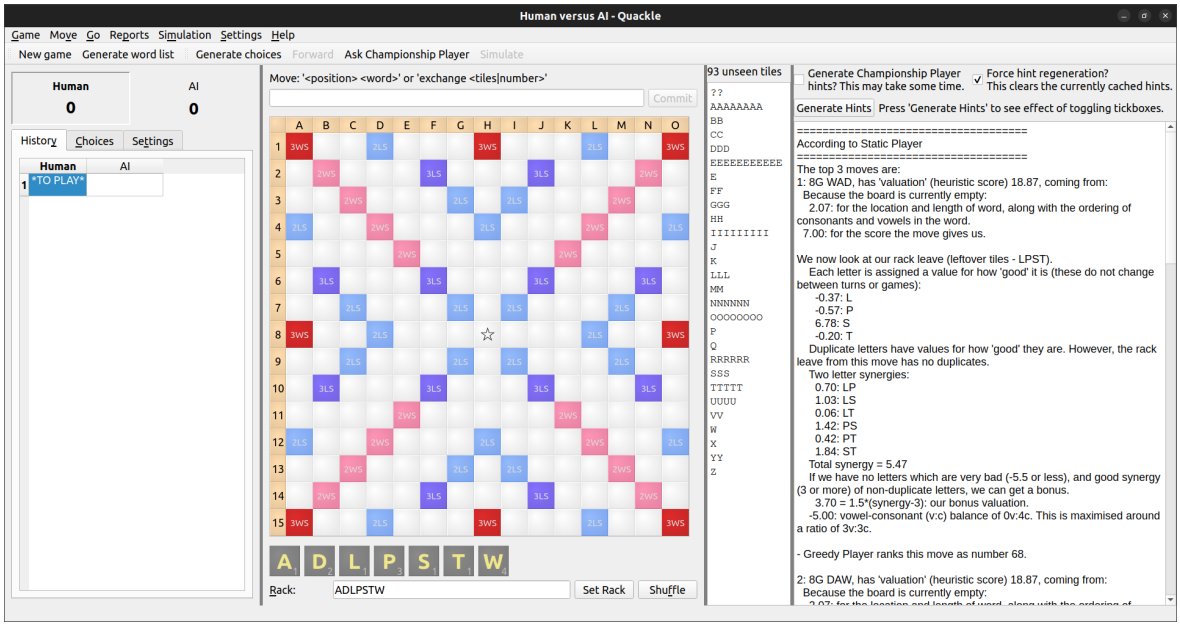

Figure 5.15: T suggested a redesign which eliminates this empty space by splitting the space on the eight vertically instead of horizontally.

Then there was a discussion about the quality of the hints. Since they were also unsure about what 'rack leave' meant, the text 'leftover tiles' was also added. They also disliked the use of the phrase "how 'good' it is". They suggested for it to be changed to something which is more specifically telling the user that these numbers factor into the calculation of valuation.

Just like in P's feedback, in the development stage, it was not obvious to T that the valuation of individual or pairs of letters stays the same between games. This is another reason why this change was implemented before they were interviewed. However, this time, it was not obvious to them that the valuations for all single letters, duplicate letters and two-letter synergies are also fixed between games. This is an important fact for the player to know, as it helps with being able to reason about other moves

they might be considering.

Similarly to P, T has the issue of not having a wide enough lexicon to play Scrabble well. They are even frustrated by how it feels like many of the words which the AIs suggest or play "are outside of the normally used dictionary". Perhaps if the goal is to only learn strategy, giving the user the option to use their own lexicon of words would not be a bad idea. This would not help them to improve at it comprehensively, but it might aid them in learning the strategy involved. This would not affect hint generation as the AIs involved only use the words which the lexica allow them to, so no other changes would be required. However, since T mentioned "dictionary", an alternative was suggested to them: what if for each word formed on the board and in the hints, they were able to hover over it and see a dictionary definition for it? They said that this would make the process of learning words "much better", as otherwise, they are "meaningless". Hence, this would also improve the program as a Scrabble play learning tool. T would almost always make use of this tool.

They did agree that over time, they would be able to learn the valuations of different rack leaves or at least general trends. If a player can 'perfectly' memorise these values (about 700 of them), then they would be able to evaluate any play just like Static and win even at a high level. They thought that this program is an improvement over just simply playing Scrabble on a physical board. They said that it is also an improvement to the original Quackle program which can generate moves for you to make, but without any explanation as to why those moves are good.

Sadly, discussion about how much they would use Championship hints did not come up naturally, so it was asked directly. They said that "out of curiosity, many times I would be interested to know".

Changes made after feedback from T: layout of visual elements altered (5.15); added extra explanation for 'rack leave' (5.16); changed phrasing of "how 'good' it is" to talk about contribution to valuation and explanation that valuations of single letters, duplicate letters and two letter synergies do not change between games improved (5.17);

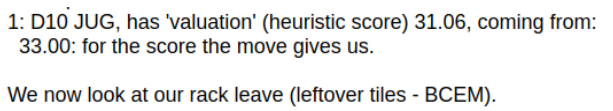

Figure 5.16: 'rack leave' is now explained to be the 'leftover tiles' in the hints

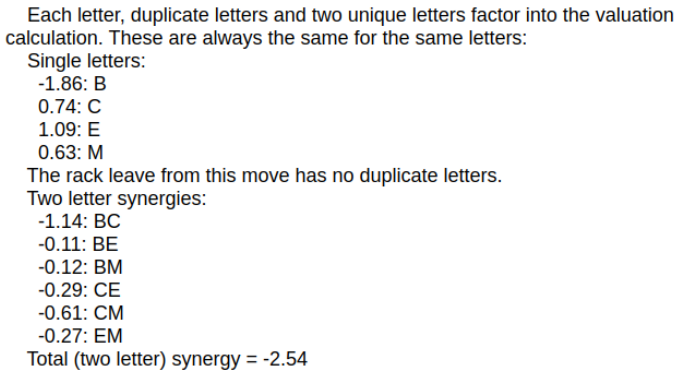

Figure 5.17: Formatting of rack leave hints change

In all of the four interviews above, there was a statement similar to "I did not know that was a word!" said. Although some of the participants might completely forget this word, others might remember it the next time they play Scrabble, or even look it up in a dictionary to confirm that it is a word. This seems to suggest that inadvertently, the users' lexicon can be expanded by using this program. This would probably also be true if they used the unaltered version of Quackle and played against an AI. However, since they are more likely to spend more time reading the hints and the associated word multiple times, they are more likely to remember the word in the future. This assertion might require further research.

A limitation to this evaluation method was that users were only given a limited amount of time with the software. If they were able to spend more time, not feel like every move of theirs was being watched, and afterwards asked for opinions again, other longer-term issues might have come up. This would have allowed for more real-world evaluation, and use of the program as it was meant to be used - over longer periods of time. Also since except for H, the others have not played much or any Scrabble before, they would not be familiar with the technical terms of the program. Multiple times during the feedback sessions, points came up about Scrabble rules. The aim of this program is not to teach them the rules of Scrabble, but instead, to teach them how to improve their strategy.

Perhaps another point that might have come up if the users were allowed longer time with the program was that after every turn, they have to press the 'Generate Hints' button. They might prefer to not need to press that button, and for it to automatically trigger on their turn. This was an active choice made since eventually, the user should be able to think without the hints. It also allows users to have the chance of trying to play well by themselves.

We can see the incorporation of all feedback in figure 5.18.

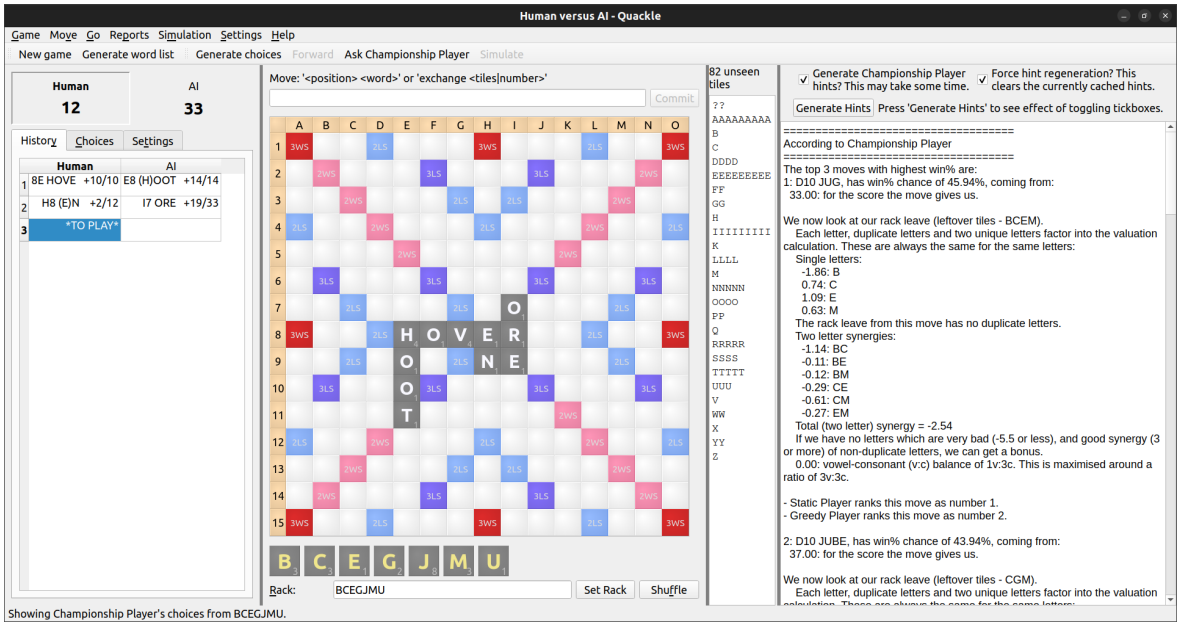

Figure 5.18: Appearance of program after incorporation of user feedback

Chapter 6

Future Work and Conclusions

According to what we have seen in the evaluation above, it seems that this project has overall been successful. Users can get feedback on plays they make, and they can be trained by a high-calibre AI - Static. The hints that Static presents are readable and can be understood by a human, which since these values are mostly AI training data, can sometimes be difficult to achieve.

There are still some aspects that could be improved upon, or require further or more in-depth research.

6.1 Future Work

6.1.1 Software Improvements

There are multiple 'quality of life' improvements that can be made to the developed software (at least to the part that deals with generating hints - possible improvements to the pre-existing code of Quackle are ignored):

- Allowing the player to choose how many top moves are shown; currently, Static and Championship display their top 3 moves and Greedy its top 5 moves, and this cannot be altered by a user. It is a fixed number. However, some users might want to see a different amount of items to read more hints and understand more quickly, or perhaps if they are already more familiar with the system, they might only want one hint.

- Allowing the user to choose exactly which AIs generate hints and which don't; currently, Static and Greedy always generate hints, and only Championship can be toggled on or off. For similar reasoning as the previous point, it might be good to allow the player finer control over which AIs generate hints and which don't.

- Having a clickable option to take you to a certain AI's hints. Instead of the user having to scroll up and down the list of all hints, it would be nice if they could just click which AI's hints they want to be taken to, and it auto-scrolls them to those hints. This just helps with navigation through the hints, as some can be longer than others.

- Making hints clickable similarly to the choices tab - clicking on a hint places the word on the board, ready to be committed as shown in 6.1. However, this might have the undesirable effect of the player being less likely to read the hints; so perhaps also adding an option to enable or disable this could also be beneficial.

- Greedy AI could be improved to only swap non-blank tiles when it can't play anything. This didn't seem like a worthwhile time investment because as we saw in the evaluation (5), Greedy was shown to be the worst of the three AIs. The situation in which Greedy swaps its entire rack is also relatively rare.

- If we look at CPU usage during Championship hint generation, it seems that only 2 cores are being used. Perhaps if a way was figured out to increase the number of cores which can be utilized, time spent on hint generation can be decreased, and hence make Championship's hints more enticing.
- Making the hints less verbose as a setting. This would allow less text to be shown on the hints display, perhaps making it more easily digestible, or at least a user can get more information at a glance. This would require the user to have experience with the full hints, as it would most likely have to make use of acronyms, or hints that only display numbers with very little to no text. To that end, being able to hover over the shortened hints might need to perhaps show what the meaning of the shortened version is, or at least display the full non-shortened hint.
- Having an option to toggle dictionary definitions of words. This can help a user in learning and remembering new words more easily. At the current stage of the program, these might be most suitably displayed in Greedy's hints as they don't have a long explanation, and adding a dictionary definition of a word might make them about equivalent to the other AIs. It is also most suitable here since Greedy is usually the one which generates the longest words, which are also usually the less known words.

Figure 6.1: Clicking on a move in the 'Choices' tab places that word down on the board - in this case, 'ZAIRE'.

6.1.2 Research

Perhaps the most useful improvement to hint generation would come from being able to generate hints from Championship. Analysing Championship would be difficult since it does hundreds of simulations to find its best play. This is not something feasible for a human to do. Perhaps a probabilistic method could be used to infer the same results by checking the probabilities of a rack to be dealt to the players. There are mentions in the code of using an 'inference engine' to infer what the opponent's rack might contain based on their plays, however, there is no implementation of it - only 'TODO' style comments.

This is partly addressed by Romero et al. (2009, [4]), where they gather data from many games and evaluate the probability of letter bring on the opponent's rack given their play.

The focus of this project was on hint generation for a two-player Scrabble game. However, Scrabble can be played with up to 4 players. investigating hint generation for games involving more than two players would be interesting as it would give rise to issues such as never being in a game of full information. Many two-player Scrabble games are decided in the end-game when both players know the opponent's exact rack. This potentially allows for perfect play, as it is no longer a game of luck. However with three or four players, this is not the case, so strategies which involve counting the points of the opponents' racks are not as valid. Interestingly, Static is agnostic as to the number of players until the end-game due to not thinking ahead. In the end-game, the only information it uses from the opponent is the total rack score (the sum of the score of each tile on a rack). This could perhaps be changed to average out over the number of opponents to achieve a similar result, as the user still knows which tiles they haven't seen. This would however need to be tested, especially since the total rack score of one player could be much higher than another's, leading to a much worse performing Static AI and hence hint generation.

The final program would allow for a Greedy-like human player to improve to a Static-like human player. However, since newer players' lexica are not as developed, it is unlikely to be able to help them improve by huge amounts. It would be of interest if a tool could perhaps be developed which could help a player learn both strategy and words simultaneously effectively.

This research was carried out on a typical Scrabble board (1.1). It would also be interesting to know how board layouts (larger or smaller boards, multiplier spot placement on the board, symmetry or asymmetry of multipliers on board) could affect gameplay, AI performance and hence also hint generation.

The AIs used here do not account for the state of the board. This includes how 'open' the board is, i.e. how easy it is to form a bingo on the current state of the board or how easily accessible multipliers are. Having AIs which also account for this could be better players, and hence better at generating hints for the user, as the user would get another datapoint to base their plays on. However, in section 3.4.1 of Sheppard (2002, [14]), they suggest that this does not seem to considerably affect the ranking of a move.

Lastly, this project used the default lexicon included in Quackle - NWL2018. It would be interesting to know how using different lexica affects how well the AIs play and hence the quality of their hints. This applies to other English lexica, but also other languages.

6.2 Conclusions

Although the Static AI is not the best for Scrabble play, it is very close. Aiding players in understanding its plays is a very good starting point to learning Scrabble strategies. It is a very good starting point as well - if Championship hints were implemented on top of Static hints, users wouldn't need to just wait around for those hints. They would be able to read the Static hints, making it less likely that they get bored. In the end, they would end up with possibly two different plays which they can compare and see why perhaps Championship's is better based on the given hints.

Any player who would like to be a high-calibre Scrabble player would have to train by playing many games. Playing against an AI is faster than playing against a real human - AIs make their moves faster than humans, so training can take less time. It is also more reliable, as when one person want to train, they don't have to wait for someone else to also join them, which could have scheduling difficulties. As we also saw in the evaluation, users would learn faster when using this project's software compared to original Quackle.

The author hopes that the end-result of this research ameliorates any reader's Scrabble skill and knowledge.

Bibliography

[1] A. W. Appel and G. J. Jacobson. The world's fastest scrabble program. *Communications of the ACM*, 31(5):572–578, 1988.

[2] Erik Arneson. Exchanging tiles in scrabble. access date: 2022/08/16 https://www.thesprucecrafts.com/scrabble-when-can-i-exchange-tiles-410944.

[3] Jason Katz Brown. Quackle, Jul 22 2019. http://people.csail.mit.edu/jasonkb/quackle/.

[4] A. Gonzalez Romero, R. Alquezar, A. Ramirez, F. Gonzalez Acua, and I. Garcia Olmedo. Human-like heuristics in scrabble. *Frontiers in Artificial Intelligence and Applications*, 202(1):381–390, 2009.

[5] S. A. Gordon. A faster scrabble move generation algorithm. *Software: Practice and Experience*, 24(2):219–232, 1994.

[6] Jason Katz-Brown and John O'Laughlin. Quackle - how it works. http://people.csail.mit.edu/jasonkb/quackle/doc/how_quackle_plays_scrabble.html.

[7] Jason Katz-Brown, John O'Laughlin, John Fultz, Matt Liberty, and Anand Buddhdev. Quackle github repository. https://github.com/quackle/quackle.

[8] Seth Lipkin and Keith Smith. Highest human rating. At the time of writing (2022/08/16), the player with highest rating of 2182 is Adam Logan https://www.cross-tables.com/results.php?p=3522.

[9] Seth Lipkin and Keith Smith. Maven rating. access date: 2022/08/16 https://www.cross-tables.com/results.php?p=10197.

[10] Seth Lipkin and Keith Smith. Quackle rating. access date: 2022/08/16 https://www.cross-tables.com/histratings.php?p=17892.

[11] F. Di Maria and A. Strade. An artificial intelligence that plays for competitive scrabble. In *Popularize Artificial Intelligence*, volume 860, pages 98–103, 2012.

[12] A. Ramírez, F. G. Acuña, A. G. Romero, R. Alquézar, E. Hernández, A. R. Aguilar, and I. G. Olmedo. *A scrabble heuristic based on probability that performs at championship level*, volume 5845 LNAI. Springer, 2009.

[13] M. Richards and E. Amir. Opponent modelling in scrabble. In *IJCAI International Joint Conference on Artificial Intelligence, pp. 1482-1487*, 2007. https://www.scopus.com/inward/record.uri?eid=2-s2.0-70549084537&partnerID=40&md5=c75d29da7aec643ed6a4432091bb403c.

[14] B. Sheppard. World-championship-caliber scrabble. *Artificial Intelligence*, 134(1-2):241–275, 2002.